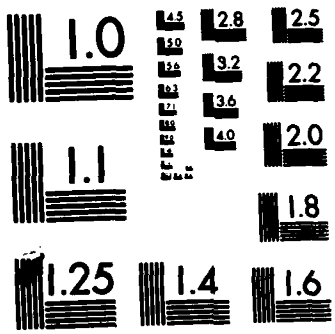
AD-A138 082 STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. H INANLI 1/3

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-32 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

STUDY OF FINITE WORD LENGTH EFFECTS
IN SOME SPECIAL CLASSES OF
DIGITAL FILTERS

THESIS

Harun Inanli
1st Lt, Turkish Air Force

AFIT/GE/EE/83D-32

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/EE/83D-32

STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS

THESIS

Harun Inanli
1st Lt, Turkish Air Force

AFIT/GE/EE/83D-32

AFIT/GE/EE/83D-32

STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Harun Inanli

First Lieutenant, Turkish Air Force

December 1983

Preface

The purpose of this thesis was to simulate some classical and innovative digital filter structures. The effect of finite word length limitations in the amplitude response of various digital filters was investigated. Also, a comparison of the result included by response and sensitivity will be discussed.

This report develops the theory of 12 different digital filter structures. Six of them, which are FIR (Finite Impulse Response) digital filters, are chosen for simulation. Anyone who is interested in the finite word length effects of these digital filter structures should find the computer programs in Appendices B, C, and D to be useful.

I want to thank my advisor, Dr Vaqar Syed, who has given me timely guidance essential to the completion of this study. A special thanks is also expressed to my committee members, Dr Tom Jones and Lt Col John Carnaghie, for their expert advice. Finally, a thank you is extended to all the students and staff of the AFIT Digital Signal Processing Laboratory for their technical support.

Harun Inanli

Table of Contents

List of Figures

Figure		Page

List of Tables

List of Symbols

$x(n)$	Input sequence
n, m, k	Integer number
$y(n)$	Output sequence
T	Transformation operator or sampling time
α	Constant
$x(z)$	Input sequence in z-transform
z	z-plane parameter
σ	Real part of z
ω	Imaginary part of z
C	Counterclockwise closed contour
Z	Transformation operator to z-domain
a, b, c	Constant
$h(n)$	Linear time invariant filter impulse response
$*$	Convolution
$u(n)$	Unit impulse response
e	2.73 or error between actual and ideal output response
S	S-plane parameter
ω_s	Sampling frequency
a_k , b_k	Digital filter coefficients
N, M	Number of poles and zeros, respectively
$H(z)$	Digital filter transfer function in z-domain
α_i	System parameter
s	Sensitivity operator
$\hat{a}_k$, $\hat{b}_k$	Quantized digital filter coefficients

$\hat{h}(k)$	Quantized linear time invariant filter impulse response
$\lvert E(e^{j\omega})\rvert_D$	Error in frequency response for direct form
$\lvert E(e^{j\omega})\rvert_C$	Error in frequency response for cascade form
e_k	FIR nested filter coefficient
NS	Nested Structure
$\hat{y}_{exp}(\cdot),\hat{y}_{act}(\cdot)$	Expected and actual quantized output, respectively
$b_s, x_s(1)$	Scaled coefficient and input, respectively
e_s	Scaled nested filter coefficient
$\hat{b}_s, \hat{x}_s(\cdot)$	Scaled and quantized coefficient and input, respectively
FFT	Fast Fourier Transformer

Δa_k , Δb_k	Error quantities in digital filter coefficients
$\hat{H}(z)$	Actual digital filter transfer function
$\hat{y}(n)$	Actual filter output sequency
$\sigma_{\Delta H}^{2}$	Variance of ΔH
q, α	Quantization step
t	Number of bits
$p(\cdot)$	Probability density
$E(\cdot)$	Mean
μ, υ	Number of nonzero coefficient
$H_i(z)$	Second order digital filter transfer function
$\hat{H}_i(z)$	Actual second order digital filter transfer function
N	Number of second order section
$\sigma_{\Delta H_D}$	Error variance for the direct form
$\sigma_{\Delta H_C}$	Error variance for the cascade form
$\sigma_{\Delta H_P}$	Error variance for the parallel form
c_k , d_k	Nested structure digital filter coefficient
p	Permutation parameter
r	Rounding operation
ε_k	Rounding error
E_{b_k} , E_{a_k}	The error in coefficient b_k and a_k , respectively
$\sigma_{\Delta H_{ND}}$	Error variance for nested form
$\sigma_{\Delta H_{NC}}$	Error variance for cascade-nested form
$\sigma_{\Delta H_{NP}}$	Error variance for parallel-nested form

Abstract

One of the main problems in digital filter implementation is that all practical devices are of finite precision. Therefore, the finite word length effect of digital filters is an area of high interest.

There are various types of digital filter structures. Due to the effect of finite word length registers, each digital filter structure gives a slightly different output response for the same transfer function. Therefore, it is important to find the best filter structure which has the lowest affect on the output response for the same transfer function.

In this paper, six IIR (Infinite Impulse Response) digital filters and six FIR (Finite Impulse Response) digital filters are investigated, theoretically, for the low sensitivity due to a finite word length register. In addition, the six FIR digital filters are simulated by computer to obtain practical results. Finally, it will be shown that NS (Nested Structure) digital filters produce the "best" response if minimum sensitivity is the figure of merit.

STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS

I. Introduction

A digital filter is a system which is used to process discrete time signals. The filter can take one of the two forms. In one form, the filter could be simply a numerical signal processing algorithm, which can be implemented on a general purpose or a special purpose digital computer. In the other form, the filter could be a dedicated piece of hardware, specially designed to fit a particular processing scheme. The choice of one form over the other involves several considerations. For example, the computer implementation is the most flexible one of the above two schemes. A simple program change is all that is required to implement a different filter. As to be expected, a hardware implementation is not as flexible. On the other hand, a digital computer implementation is inherently slower than the hardware implementation. Furthermore, hardware implementation may be cheaper in terms of hardware cost, but more expensive in terms of development cost. No matter which particular form is chosen, the so-called "finite word length effects" should carefully be taken into account for any useful implementation of a digital filter. These effects stem from the

fact that any digital computer or digital network operates with finite number of bits. Thus, signal quantization, filter coefficient quantization, and register overflows must be expected. Depending upon what particular structure one wants for a filter implementation, these effects, commonly called the "finite word length effects," will result in significantly different filter responses.

A desirable implementation of a digital filter is the one that minimizes the effect of finite word length on the filter performance. We will term such an implementation the "low sensitivity realization." The main purpose of this study will be to examine from literature, various low sensitivity structures, analyze bounds on their performance and present a comparison of these realizations in terms of coefficient sensitivity and round-off errors. The work presented here will be based on computer simulation of digital filters using register lengths of variable number of bits and the finite precision arithmetic.

Scope of This Study

This study involves both theoretical and experimental investigations. The main goal of this thesis is to implement typical digital filters of the low-pass, band-pass, and high-pass type using various structures reported in literature. Then, taking into account the finite word length limitations of digital machines, the filter will be theoretically analyzed for register overflows, amplitude response errors, and limit

cycling (if any). These theoretical predictions will be compared with digital filters of various word lengths simulated on the digital computer in the AFIT Digital Signal Processing Laboratory.

Organization of This Thesis

This thesis has been organized as follows. Following this introduction chapter, Chapter I, we present in Chapter II a brief review of the theory, terms and definitions that pertain to digital filters. Included here will be the finite impulse response (FIR) and infinite impulse response (IIR) filters, digital filter realizations, number systems and their properties.

In Chapter III, some recently reported and some commonly known structures for the realization of digital filters, both for IIR and FIR filters, will be reviewed. Various issues related to the finite word length of digital systems will be described here. Furthermore, a sensitivity analysis of the various filter structures described in this chapter will be presented along with theoretical upper bounds on their performance and limit cycling (if any) due to the round-off noise effects.

In Chapter IV, simulation examples of the digital filter structure described in Chapter III will be presented.

Finally, in Chapter V, a conclusion of this study will be presented, and possible directions for future work on this subject will be outlined.

II. Digital Filter Preliminaries

Introduction

A digital filter can be represented by a network which contains a collection of interconnected elements. Analysis of a digital filter is the process of determining the response of the filter network to a given input.

This chapter is an introduction to the basics of digital filters. A brief review of basic definitions, terminology and mathematical preliminaries related to the digital filter will be presented here.

The Digital Filter As A System

A digital filter can be defined as an operator which *transforms an input sequence* $x(n)$, $n=0, \pm 1, \pm 2, \pm 3 \ldots$, into an output sequence $y(n)$, written symbolically as

$$\{y(n)\} = T\{x(n)\} \tag{2-1}$$

where T is the transformation operator. We will be concerned here with the class of operators which are termed linear and shift invariant. An operator T is linear if the principle of superposition holds; i.e., if

$$\{y_1(n)\} = T\{x_1(n)\}$$

and

$$\{y_2(n)\} = T\{x_2(n)\}$$

then

$$\{\alpha_1 y_1(n) + \alpha_2 y_2(n)\} = T\{\alpha_1 x_1(n) + \alpha_2 x_2(n)\} \qquad (2\text{-}2)$$

where α_1 and α_2 are constant.

An operator T is shift invariant if a shift of m in the input sequence $\{x(n)\}$ produces the same shift m in the same direction in the output sequence $\{y(n)\}$. That is,

$$\{y(n-m)\} = T\{x(n-m)\} \qquad (2\text{-}3)$$

A digital filter satisfying the properties defined by Equations (2-2) and (2-3) above is called a linear shift-invariant digital filter.

A more restricted class of linear time invariant digital filter can be defined by imposing causality and stability. A causal system is the one for which the output for any $n=n_o$ depends on the input for $n \le n_o$ only; i.e., if the input sequences $x_1(n)$ and $x_2(n)$ are such that

$$x_1(n) = x_2(n) \quad \text{for} \quad n \le n_o$$

and

$$x_1(n) \neq x_2(n) \quad \text{for} \quad n > n_o \qquad (2\text{-}4)$$

then, the output sequences $y_1(n)$ and $y_2(n)$ are related as

$$y_1(n) = y_2(n) \quad \text{for} \quad n \le n_o \qquad (2\text{-}5)$$

This is illustrated in Figure 1.

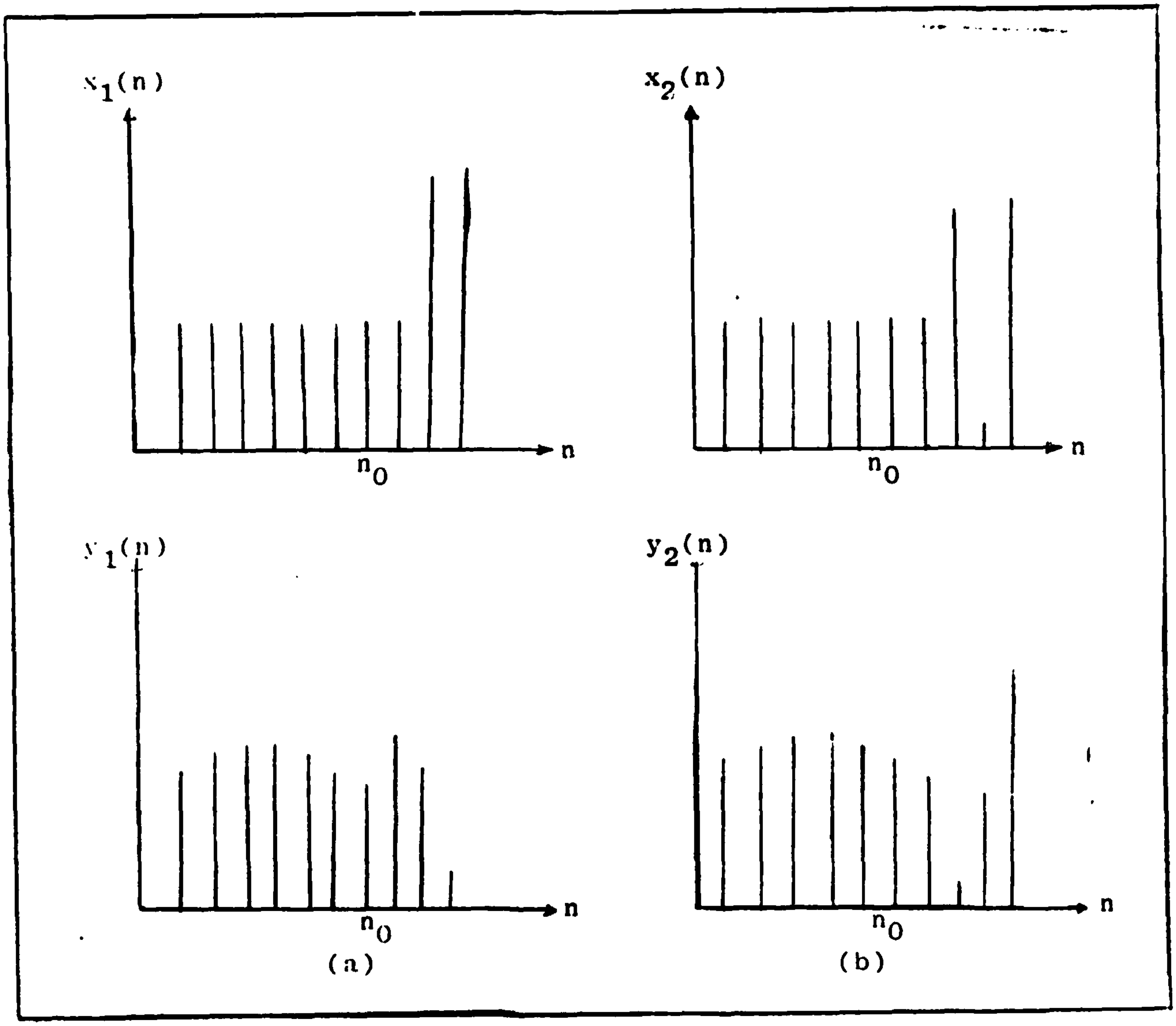


Figure 1. Illustration of Causality: (a) Response to $x_1(n)$, (b) Response to $x_2(n)$

A stable system is one for which every bounded input produces a bounded output. In this study, we will only consider causal and stable digital filters. Furthermore, without

loss of generality, we will assume that the input to the digital filters discussed in this thesis are sampled time-domain signals, and that the outputs are also sampled time-domain signals specified at the sampling instants nT, $n = 0, \pm 1, \pm 2, \ldots$. Thus, instead of the nomenclature "shift-invariant," we will use "time-invariant." Furthermore, we will assume that the sampling rate employed satisfies the Nyquist criterion given by the following statement of the sampling theorem.

The Sampling Theorem. A band limited signal having no spectral components above a frequency of B Hz is determined uniquely by its values at uniform intervals spaced no more than $\frac{1}{2B}$ second apart.

For proof, the reader is referred to [1] or [2].

Fundamental to the design of linear, time-invariant digital filters is the Z-transform concept. We, thus, briefly review the essentials of the Z-transforms.

The Z-Transform

The two-sided Z-transform $X(z)$ of a sequence $x(n)$ is defined as

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \tag{2-6}$$

where z is a complex variable of the form $z = \sigma + j\omega$.

If the summation proceeds for $n \geq 0$, we have the one-sided Z-transform $X_1(z)$ defined as

$$X_1(z) = \sum_{n=0}^{\infty} x(n)\, z^{-n} \tag{2-7}$$

The infinite series of Equations (2-6) and (2-7) does not always converge. However, we assume that for the sequences of interest here, the series do converge.

If the Z-transform of a sequence $x(n)$ exists, then the sequence $x(n)$ can be recovered from $X(z)$ via an inverse operation called the inverse Z-transform, given by

$$x(n) = \frac{1}{j2\pi} \oint_C X(z)\, z^{n-1}\, dz \tag{2-8}$$

Here, C is a counterclockwise closed contour in the region of convergence of $X(z)$, and encircles the origin of the Z-plane. The details of the contour integration of Equation (2-8) are outlined in [3] and [4].

A few properties of the Z-transforms and the relationship of the Z-plane with the S-plane which will be useful in the subsequent development are reviewed next.

(a) Linearity. Consider two sequences $x(n)$ and $y(n)$, with Z-transforms $X(z)$ and $Y(z)$ respectively; i.e., symbolicly,

$$Z[x(n)] = X(z)$$

and

$$Z[y(n)] = Y(z)$$

then, for constants a and b

$$Z[ax(n) + by(n)] = aX(z) + bY(z) \qquad (2\text{-}9)$$

(b) Shift. Consider a sequence $x(n)$ such that

$$Z[x(n)] = X(z)$$

then

$$Z[x(n\pm m)] = z^{\pm m} X(z) \qquad (2\text{-}10)$$

Thus, for example, for constants a, b, and c

$$Z[ax(n) + bx(n-1) + cx(n-2)] = aX(z) + bz^{-1} X(z) + cz^{-2} X(z)$$

(c) Convolution of Sequences. The convolution sum of two sequences $x(n)$ and $h(n)$ is defined by the following two equivalent summations:

$$\sum_{k=-\infty}^{+\infty} x(k)\, h(n-k)$$

$$\sum_{k=-\infty}^{+\infty} x(n-k)\, h(k) \qquad (2\text{-}11)$$

If, for a linear time invariant filter, $h(n)$ and $x(n)$ represent its impulse response and input, respectively, then its output $y(n)$ is given by the above two summations. Denoting the convolution by *, we then write

$$y(n) = x(n) * h(n) \tag{2-12}$$

Convolution in the time domain is equivalent to the multiplication in the Z-domain. Thus

$$Y(z) = X(z)\,H(z) = H(z)X(z) \tag{2-13}$$

where

$$Y(z) = Z[y(n)]$$

$$X(z) = Z[x(n)]$$

$$H(z) = Z[h(n)]$$

(d) Initial Value Theorem. If $\lim_{z\to\infty} X(z)$ exists and $x(n)$ is zero for $n<0$, then

$$x(0) = \lim_{n\to 0} x(n) = \lim_{z\to\infty} X(z) \tag{2-14}$$

For example:

$$x(n) = u(n)\left[\frac{1}{3} + \frac{2}{3}\left(-\frac{1}{2}\right)^{n}\right]$$

where $u(n)$ is the unit step. The Z-transform of $x(n)$ is

$$Z[x(n)] = X(z) = \frac{(2z-1)z}{2(z-1)(z+0.5)}$$

Initial value in time-domain and Z-domain are

$$\lim_{n\to 0} x(n) = 1$$

$$\lim_{z \to \infty} X(z) = 1$$

So,

$$\lim_{n \to 0} x(n) = \lim_{z \to \infty} X(z)$$

(e) Final Value Theorem. If $X(z)$ converges for $|z|>1$ and all the poles of $(1-z)X(z)$ are inside the unit circle, then

$$\lim_{n \to \infty} x(nT) = \lim_{z \to 1} [(1-z^{-1})X(z)] \qquad (2\text{-}15)$$

Mapping to the Z-Plane. The relationship between points in the Z-plane and the S-plane is described by

$$z = e^{Ts} \qquad (2\text{-}16)$$

where

$e = 2.73$

T = sampling time

z = Z-plane parameter

s = S-plane parameter in the complex form of $\sigma + j\omega$

The transformation can be investigated by inserting $s = \sigma + j\omega$ into Equation (2-16) to obtain

$$z = e^{\sigma T} e^{j\omega T} \qquad (2\text{-}17)$$

Sampling time can be found from

$$T = \frac{2\pi}{\omega_s} \qquad (2\text{-}18)$$

where ω_s is sampling frequency. Let us substitute Equation (2-18) into Equation (2-17). Therefore

$$z = e^{\sigma T} e^{j2\pi\omega/\omega_s} \qquad (2\text{-}19)$$

Equation (2-19) shows that:

1. Lines of constant σ_1 in the S-plane map into circles of radius equal to $e^{\sigma_1 T}$ in the Z-plane. Specifically, the segment of the imaginary axis σ in the S-plane of width ω_s maps into the circle of unit radius in the Z-plane. So, the condition for stability is that all roots of the characteristic equation lie within the unit circle.

2. Lines of constant ω in the S-plane map into radial rays drawn at the angle ωT in the Z-plane. The portion of the constant ω line in the left half of the S-plane becomes the radial ray within the unit circle in the Z-plane. The corresponding paths, as discussed above, are shown in Figure 2. For further detail, the reader is referred to [5].

Classification of Digital Filters

In general, linear shift-invariant digital filters are classified into two major groups:

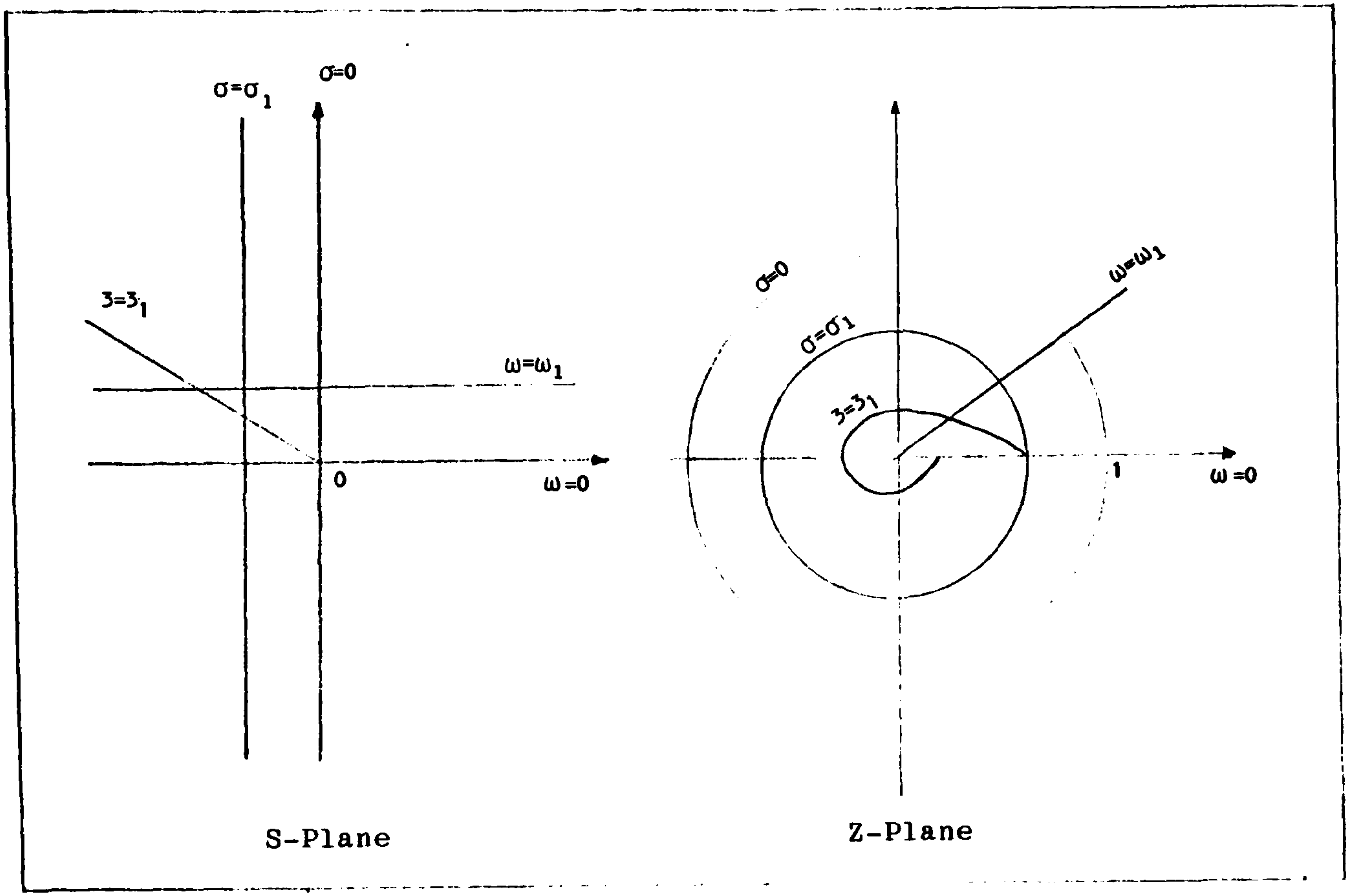


Figure 2. Transformation from the S-Plane to the Z-Plane

1. IIR (Infinite Impulse Response) filters or recursive filters.

2. FIR (Finite Impulse Response) filters or non-recursive filters.

Infinite Impulse Response Filters. A filter defined by an impulse response sequence for which the range of non-zero values extends to positive infinity, negative infinity, or both. The current output for IIR filters depends upon current and/or previous inputs as well as previous outputs. This input-output relationship satisfies the difference equation,

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \qquad (2\text{-}20)$$

where

$y(n)$ = output sequence

$x(n)$ = input sequence

a_k, b_k = digital filter coefficients

N, M = the number of poles and zeros, respectively

In the Z-domain, Equation (2-20) can be represented by its transfer function H(z), which in this case has a very simple form.

$$Y(z) = H(z)X(z) \qquad (2\text{-}21)$$

where H(z), the filter transfer function, is given by

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{2-22}$$

The roots of numerator and denominator polynomials are the zeros and poles of the filter, respectively, in Equation (2-22). The poles determine the stability of digital filters. Thus, if the poles of a digital filter are inside a unit circle in the Z-plane, the filter is stable.

<u>Finite Impulse Response Filters</u>. A filter defined by an impulse response sequence which is nonzero over only a finite range and the output is independent of previous output. In this case, the filter coefficients satisfy the following conditions in Equation (2-20)

$$a_k = 0 \quad \text{for} \quad k \neq 0 \tag{2-23}$$

The difference Equation (2-20) reduces to

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) \tag{2-24}$$

and, hence, the transfer function in Z-domain reduces to

$$H(z) = \sum_{k=0}^{M} b_k z^{-k} \tag{2-25}$$

If the above equation is multiplied by $\frac{z^M}{z^M}$, we get

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{M-k}}{z^M} \tag{2-26}$$

It is obvious from Equation (2-25) that FIR filters have only finite zeros; all the poles of these filters are located at $z = 0$.

The choice between an FIR filter and IIR filter depends on the application. High selectivity can easily be achieved with low-order transfer function in application of IIR filters by placing the poles anywhere inside the unit circle. In the case of FIR filters, this can be done only by using a relatively high order for the transfer function. In practice, the cost of digital filter tends to increase and its speed tends to decrease as the order of transfer function is increased. Hence, for high-selectivity applications, the choice is expected to be an IIR filter. However, FIR filters have two attractive properties. First, there is the possibility of designing exact linear phase, required in many applications. Second, FIR filters are never unstable. A detailed consideration about this subject is given in [6].

Realization

From Equations (2-22) and (2-25) in the previous section, it is obvious that the basic operations required for realization of these equations are additions, shift and multipliers. The interconnections of these basic operations specify the filter structure.

There are an infinite variety of structures that will result in the same relationship between the input samples $x(n)$ and the output samples $y(n)$. The selection of the filter structure is very important in design process because it directly affects the efficiency and performance of the filter. Further details of the various digital filter structures and their effect on the efficiency and performance of digital filters will be discussed as needed in the chapters that follow.

As discussed in the previous chapter, the process of quantization is fundamental to digital filters. The following section is concerned with a brief description of this important aspect of digital machines.

Quantization

After the selection of the filter class and structure, the next step is the realization of this structure via a general purpose computer or special purpose hardware. Either way, there is an inherent limitation on accuracy, because all digital networks operate with only a finite

number of bits, which in turn specify the register word length. This means that the coefficients used in implementing a given filter will, in general, not be exact, and therefore the poles and zeros of the filter will be different from the desired poles and zeros. This movement of poles and zeros causes errors in the desired output of the digital filter, and in the IIR case, may even make it unstable!

The quantization of coefficients and signal in implementing a given filter is achieved either by rounding or by truncation (chopping). We thus discuss rounding and truncation in the binary domain in the following paragraphs.

Rounding. In rounding, a one or zero is first added to the t^{th} bit (t is the number of bits in the register word length excluding sing bit) according to whether the $(t+1)$'th bit is one or zero. Then, only the first t bits of the results are kept. For example, let us assume arbitrary number for coefficients or signal $a = 0.234$ and the register word length $t = 7$. The binary representation of this number is 0.001110111. Since the word length is limited to seven bits and the 8^{th} bit is a one, one is to be added to the 8^{th} bit of numbers. Then, the result is 0.0011110. So, the number will be realized as 0.0011110 instead of 0.0011101111 . . . .

Truncation. In truncation, those bits beyond the most significant t bits are simply dropped. Thus, in the above example, the number used in rounding will be realized

as 0.0011101 if computations are based on truncation technique.

The error resulting from number quantization will change the desired input and filter coefficient. This error can be classified in various categories as follows:

1. Input-quantization errors
2. Coefficient-quantization errors
3. Product quantization errors.

In addition the word length, the accuracy of a digital filter depends on two important factors: (1) the type of arithmetic used, and (2) as stated before, the form of realization.

Number Representation

Before studying the error behavior of digital filters, it is necessary to describe how the numbers, used in the implementation, are represented. The implementation of digital filter is based on the binary number representation. Binary number is represented as a string of binary digits (bits) that are either zero or one with a binary point dividing the integer part from the fractional part.

There are two possible ways of specifying the position of the binary point in a register: one, by giving it a fixed-point position, which is known as "fixed-point binary number representation," and the other, by employing

a floating-point which is known as "floating-point number representation." In fixed-point, binary point is always fixed in one position. The two positions used are: (1) a binary point in the extreme left of the register which makes the number fraction, and (2) a binary point in the extreme right which makes the number integer. For example, let "a" be the arbitrary binary number and Δ the binary point.

$$a = \Delta\, 10110101$$

(binary point in the extreme left position)

$$a = 10110101_{\Delta}$$

(binary point in the extreme right position)

In a floating-point arithmetic, no specific physical position of the register is assigned to the binary point. The numbers need two registers. The first represents a signed fixed-point number and the second, the position of the radix point. The contents of the first register are called the coefficient or mantissa and the contents of the second register is called the exponent (or characteristic).

Floating-point is always interpreted to represent a number in the following form:

$$c.r^{e}$$

where c represents the contents of the coefficient register and e, the contents of the exponent register. For example,

the number $+1001_{\Delta}110$ can be represented as follows:

0100111000	00100
(coefficient)	(exponent)

The first bit, at the extreme left in both registers, represents the sign bit. Zero stands for positive, and one stands for negative numbers. For detailed information about the number representation, the reader is referred to [7] or [8]. This study will be based on fixed-point binary number representation, with the binary number in the extreme left of the register, representing the sign of the number.

There are many other schemes for the representation of negative numbers. The reason that this particular scheme was chosen for number representation, as we will discuss later in this chapter, is to make the handling of addition and subtraction easy. In this number representation, when the number is negative, the sign is represented by a "1" in the extreme left position of the register, and the rest of the number may be represented in any one of the following three different ways:

1. Sign-Magnitude
2. Sign-1's complement
3. Sign-2's complement

As an example, the binary number 6 is written below by using 4-bit available register in the three representations.

	+6	-6
Sign-magnitude	0110	1110
Sign-1's complement	0110	1001
Sign-2's complement	0110	1010

The "0" in the left-most bit of the register represents the positive numbers. As we can see from the above example, the representations of positive number are the same in all systems. The magnitude of sign-1's complement is obtained by exchanging 0 and 1 in sign-magnitude representation. Then, two's complement is obtained by adding 1 to the sign-1's complement. In this study, the numbers are represented by sign-magnitude. However, when they are added or subtracted, they are represented in sign-2's complement. The basic operations of shifts, additions, and multiplication are next discussed in the number system used in this thesis.

Shifts

Shift is the basic operation of binary multiplaction, and can be a shift-left or a shift-right. In any case, the sign bit should remain the same. In arithmetic, shift-left multiplies a signed binary number by 2. In arithmetic, shift-right divides the number by 2.

Addition

The addition can be done in all number systems; but the easiest way to handle the addition is sign-2's complement

addition [7]. Both augend and addend are represented in sign-2's complement and the sum is obtained in sign-2's complement also. The advantage of sign-2's complement addition over the others is that the sign bit is automatic, and thus, one does not have to worry about it. An example is shown below

```
    -9       1110111
  + -9     + 1110111
  ----     ---------
   -18       1101110
```

As we can see from the above example, including sign-bit is added and a carry in the most significant (sign) bit is discarded. For further detail about this, the reader is referred to the reference [8] or [9]. Another problem that we can run into during addition is overflow. When two numbers of n digits each are added and the sum occupies n+1 digits, we say that an overflow has occurred. There are a variety of ways of checking the overflow. In this study, we handle overflow by setting another bit after sign bit to the augend register. Let us look at an example: first without checking overflow and the second with checking overflow.

```
   -35       1011101
 + -40     + 1011000
 -----     ---------
   +53       0110101      incorrect

   -35       01011101
 + -40     +  1011000
 -----     ----------
   -75       10110101     correct
```

Multiplication

In digital filter implementation, multiplier is the device which takes most of the time. Both multiplicand and multiplier require n bit register to represent the number in sign-magnitude number system. But, the product register requires 2n bit register to get the correct result.

Multiplication of two fixed-point binary numbers in sign-magnitude representation is done with paper and pencil by successive additions and shifting. For example,

$$\begin{array}{r} 6 \\ \times\ 3 \\ \hline 18 \end{array} \qquad \begin{array}{r} 110 \\ \times\ 011 \\ \hline 110 \\ 110 \\ 000 \\ \hline 10010 \end{array}$$

The sign of the product is determined from the signs of the multiplicand and multiplier. If they are alike, the sign of the product is plus. If they are unlike, the sign of the product is minus.

In digital filter implementation, it is convenient to change the process slightly for multiplication explained above. Instead of providing digital circuits to store and add simultaneously as many binary numbers as there are ones in the multiplier, it is convenient to provide circuits for the summation of only two binary numbers and successively accumulate the partial product in a register. The previous numerical example is repeated here to clarify the proposed

multiplication process:

multiplicand	110
multiplier	011
1st multiplier bit=1 copy multiplicand	110
shift right to obtain partial product	0110
2nd multiplier bit=1 copy multiplicand	110
add multiplicand to previous partial product	10010
shift right to obtain 2nd partial product	010010
3rd multiplier bit=0, shift right to obtain the final product	0010010

We can ignore the zeros at the left hand side; thus, we can easily see that the above is the same result as we obtained with the hand calculation.

Summary

In this chapter, we reviewed a number of basic definitions related to digital systems, including realization, quantization and number systems.

The definition of digital filters, linearity, causality, and stability were presented and the z-transform was reviewed. Some theories in z-transform such as linearity, shift, convolution, initial and final value, and the relation between the s-plane and the z-plane were studied.

The two broad classes of digital filters such as FIR and IIR were considered and their comparisons were made. The definition of realization and quantization, type of

quantization, such as rounding and truncating, were outlined.

Finally, number systems such as floating point, fixed point, signed magnitude 1's complement, 2's complement and arithmetic operations such as shift, addition, multiplication, and overflow problems were reviewed.

III. Realization and Sensitivity Analysis

Introduction

The realization is the step in digital filter implementation process that converts a given transfer function into an algorithm or a network. The realization step is carried out on the assumption that the arithmetic devices to be employed are of infinite precision. Since practical devices are of finite precision, it makes the realization of digital filter more complicated.

There are various types of filter structures; and due to the effect of finite word length registers, each one of them gives slightly different output response for the same transfer function. Therefore, it is important to find the filter structure which has the lowest effect on the output response of the filter.

In this chapter, previously well-known filter structures and a recently reported new structure [13] will be discussed for both IIR and FIR systems. Considered structure are direct, cascade and parallel, as well as a newly reported structure, the so-called "Nested Structure" (NS). Along with the realization of the filter structure, the sensitivity will be analyzed. To do this, it is more convenient to consider IIR and FIR filters separately.

Direct Form

It is one of the simplest forms of realization, and can be obtained by examining Equation (2-20) for IIR and Equation (2-24) for FIR filters. Kaiser [11] has shown that the sensitivity of the filter response to the accuracy of representation of the denominator coefficients in the IIR direct form increases very rapidly with increases in filter order compared to either the cascade or the parallel form. However, in this study, it is shown that the same is not true for FIR filters.

IIR Filters. This filter is characterized by an input-output relationship of Equation (2-20), or equivalently by its Z-domain transfer function H(z), which is given by Equation (2-22). For the purpose of realization, Equation (2-22) can be written in the alternative form

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}} \tag{3-1}$$

The direct form is simply defined to be a straightforward implementation of Equation (2-20) or Equation (3-1). The corresponding digital filter structure is shown in Figure 3. Note that the direct form has the minimum number of delay elements.

FIR Filters. The input and output relationship of FIR filters is expressed by Equation (2-24), rewritten below for convenience, and labeled by (3-2).

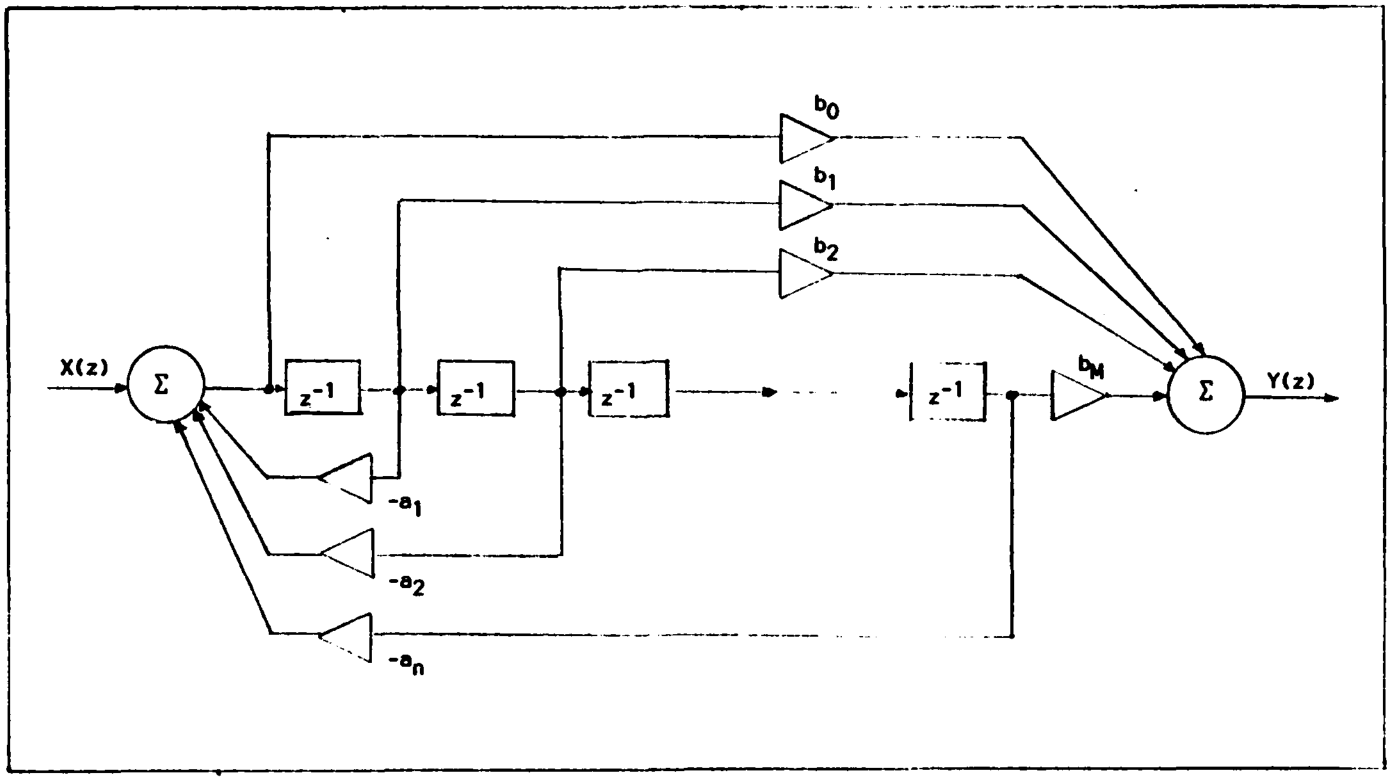


Figure 3. Direct Form for IIR Digital Filters

$$y(n) = \sum_{k=0}^{M} h(k)\ x(n-k) \qquad (3\text{-}2)$$

The transfer function H(z) in the Z-domain can then be expressed as,

$$H(z) = \sum_{k=0}^{M} h(k) z^{-k} \qquad (3\text{-}3)$$

H(z) is a polynomial in z^{-1} of degree M. Thus, H(z) has M poles at z=0 and M zeros that can be anywhere in the finite Z-plane. The structure shown in Figure 4 is simply a straightforward implementation of Equation (3-3). It is obvious that the direct form structure for FIR systems is a special case of the direct form structure for IIR systems when all the coefficients a_k of Equation (2-20) are zero.

Cascade Form

Cascade structure is obtained by factoring the numerator of the transfer function H(z), which is an n^{th} order polynomial in z^{-1}, into numerous second order factors involving the powers z^{-2}, z^{-1}, and z^{0}. Each one of these second order polynomials is then realized as a second order filter section. Cascading these sections results in the required digital filter. There is clearly considerable freedom in the choice of the ordering of these sections.

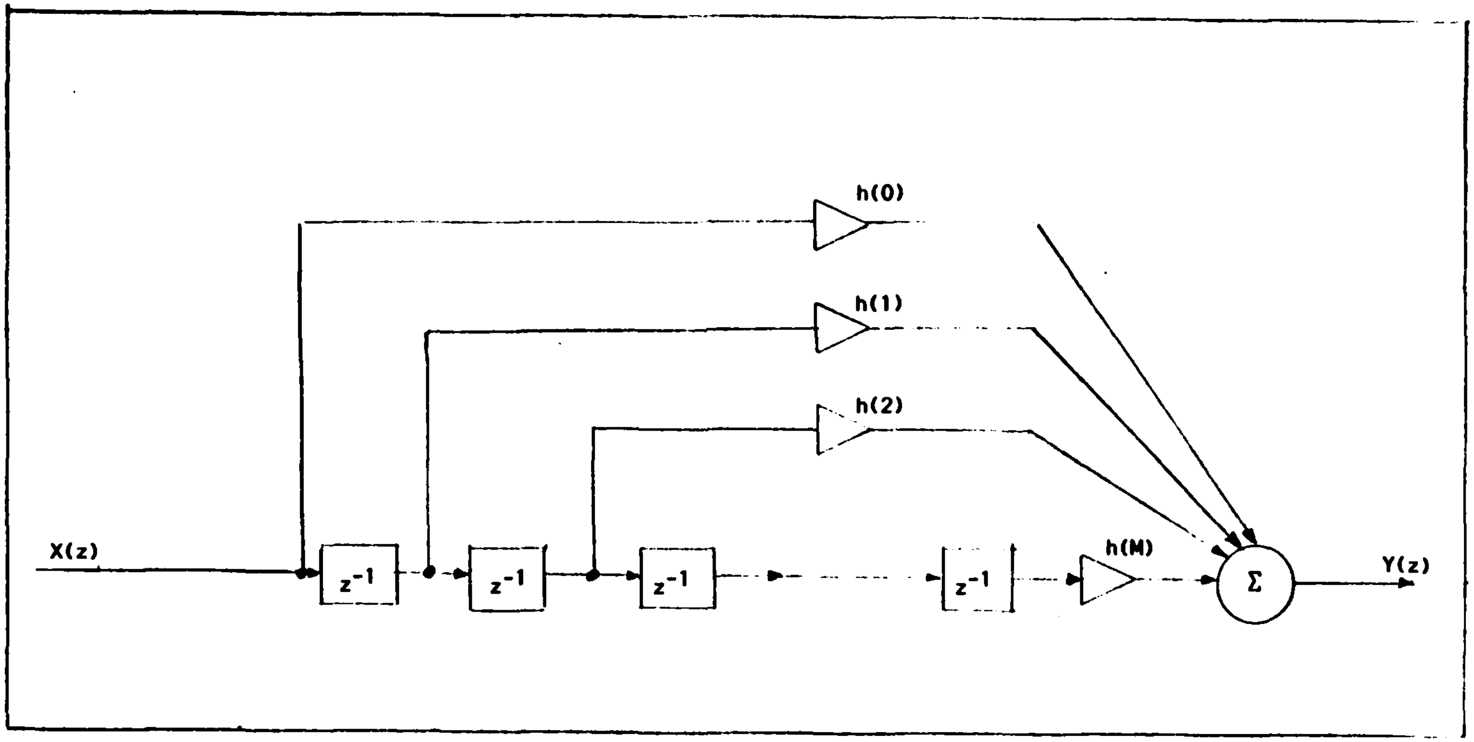


Figure 4. Direct Form for FIR Digital Filters

Cascade structure tends to have comparatively low sensitivity to the filter parameter variations [1].

IIR Filters. Digital filter transfer function H(z) expressed by Equation (2-22) can be factored into a product of second order transfer function as

$$H(z) = \prod_{i=1}^{M} H_i(z) \tag{3-5}$$

where

$$H_i(z) = \frac{b_{0_i} + b_{1_i} z^{-1} + b_{2_i} z^{-2}}{1 + a_{1_i} z^{-1} + a_{2_i} z^{-2}} \tag{3-6}$$

Each $H_i(z)$ is then realized separately. The resulting filter structure is shown in Figure 5.

There is considerable flexibility in the manner in which the poles and zeros are paired together and in the order in which the resulting second-order subsystems are cascaded. However, they have slightly different response due to the finite word length effect. We will show some examples to illustrate this phenomenon in Chapter IV.

FIR Filters. Similar to IIR filters, the digital filter transfer function H(z) expressed by Equation (2-25) can be factored into a product of second-order transfer

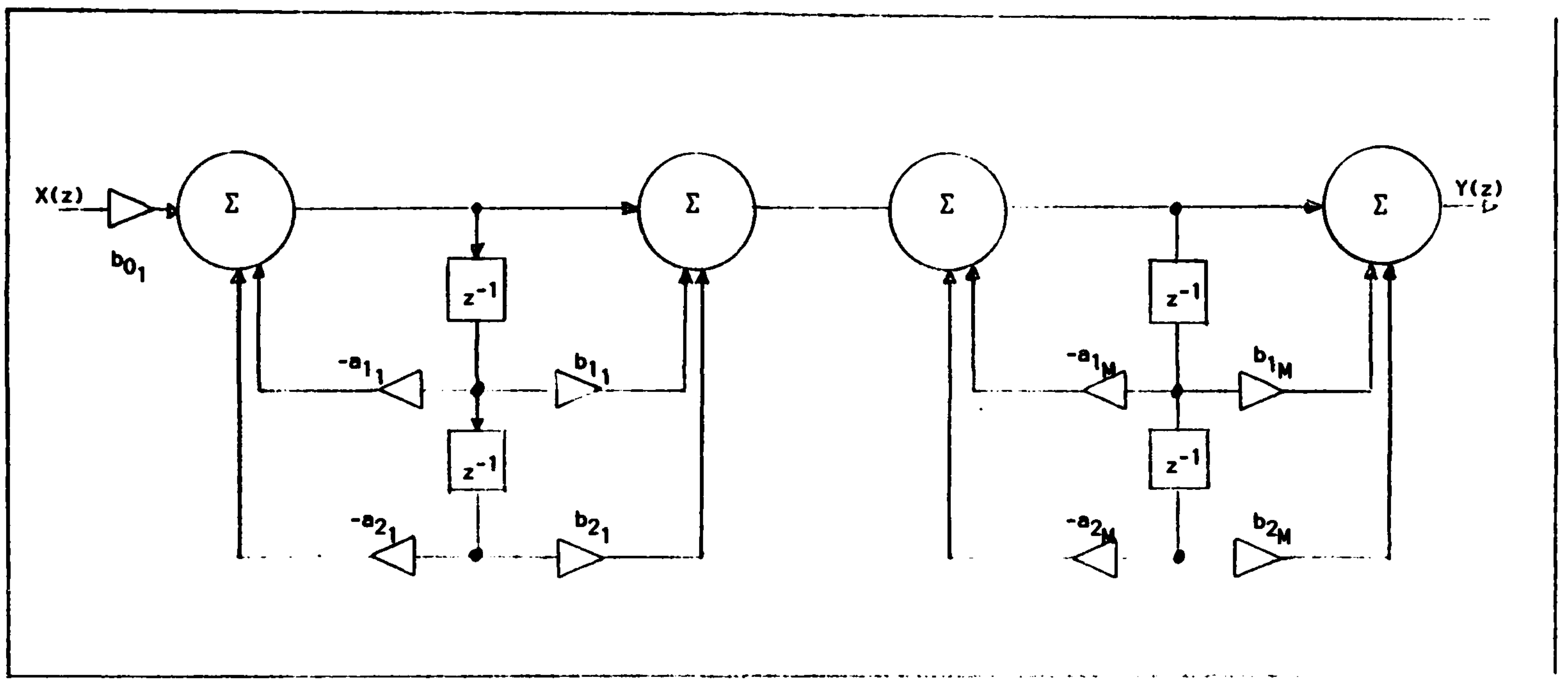


Figure 5. Cascade Form for IIR Digital Filters

function is

$$H(z) = \prod_{i=1}^{M} H_i(z) \qquad (3\text{-}7)$$

where

$$H_i(z) = b_{0_i} + b_{1_i} z^{-1} + b_{2_i} z^{-2} \qquad (3\text{-}8)$$

The corresponding filter structure is shown in Figure 6.

We have seen that each second-order section of FIR filter is the special case of the second-order section of IIR filter in which all the poles are located at z=0.

Parallel Form

One of the important parameters in digital filter implementation is the computation time required to get the output response from the given input which, in turn, depends on the operational speed of each device used between the input and the output. When the speed is important in implementation, parallel form is very convenient.

Parallel form, similar to cascade form, is obtained by partial fraction expansion of the transfer function H(z), into numerous second order factors involving the powers z^{-2}, z^{-1}, and z^{-0}. Each one of these second-order factors is then realized

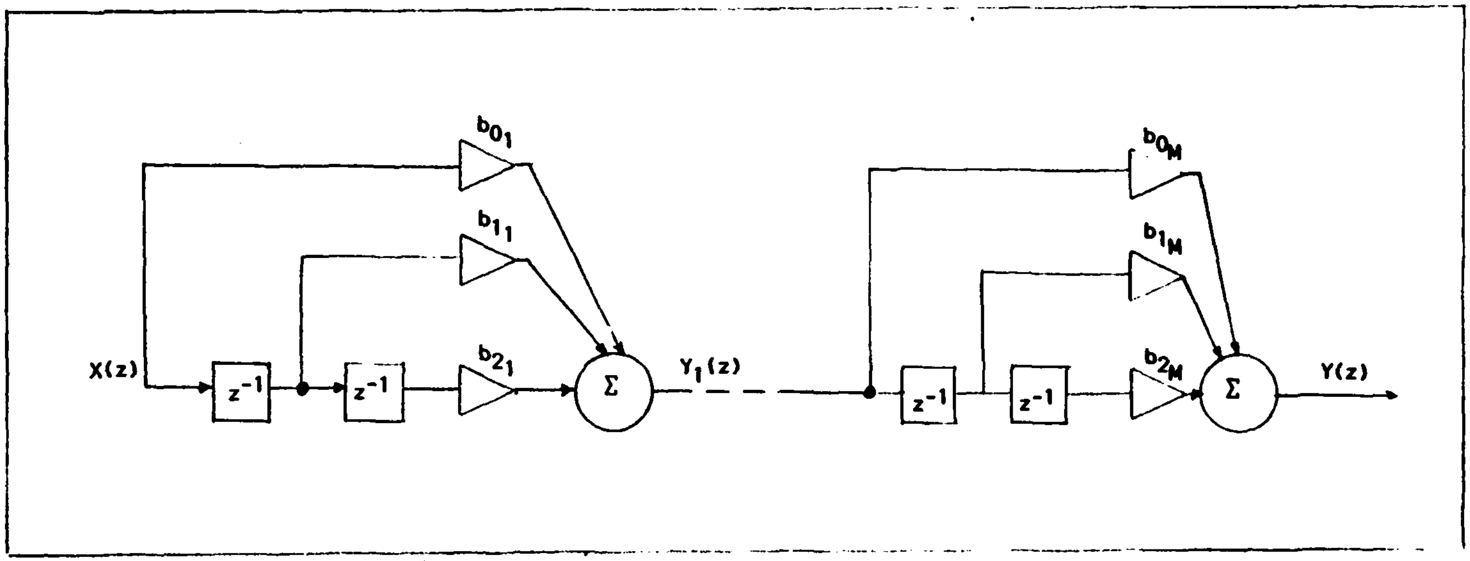


Figure 6. Cascade Form for FIR Digital Filters

as a second order filter section. Instead of cascading, connecting in parallel of these sections results in the required digital filter.

IIR Filters. Digital filter transfer function H(z) given by Equation (2-22) can be expressed as a partial-fraction expansion in the form

$$H(z) = \sum_{i=1}^{M} H_i(z) \tag{3-9}$$

where $H_i(z)$ is of the same form as given by Equation (3-6).

These second-order transfer functions $H_i(z)$ are then connected in parallel. The result is the filter structure shown in Figure 7.

FIR Filters. Digital filter transfer function H(z) given by Equation (2-25) can be expressed as a partial fraction expression in the form:

$$H(z) = \sum_{i=1}^{M} H_i(z)$$

where $H_i(z)$ is the same as Equation (3-8). The corresponding structure is shown in Figure 8.

Nested Structure

The direct form, as expressed before, is generally more sensitive to the effects of coefficient quantization in fixed-point implementation, if the dynamic range of the

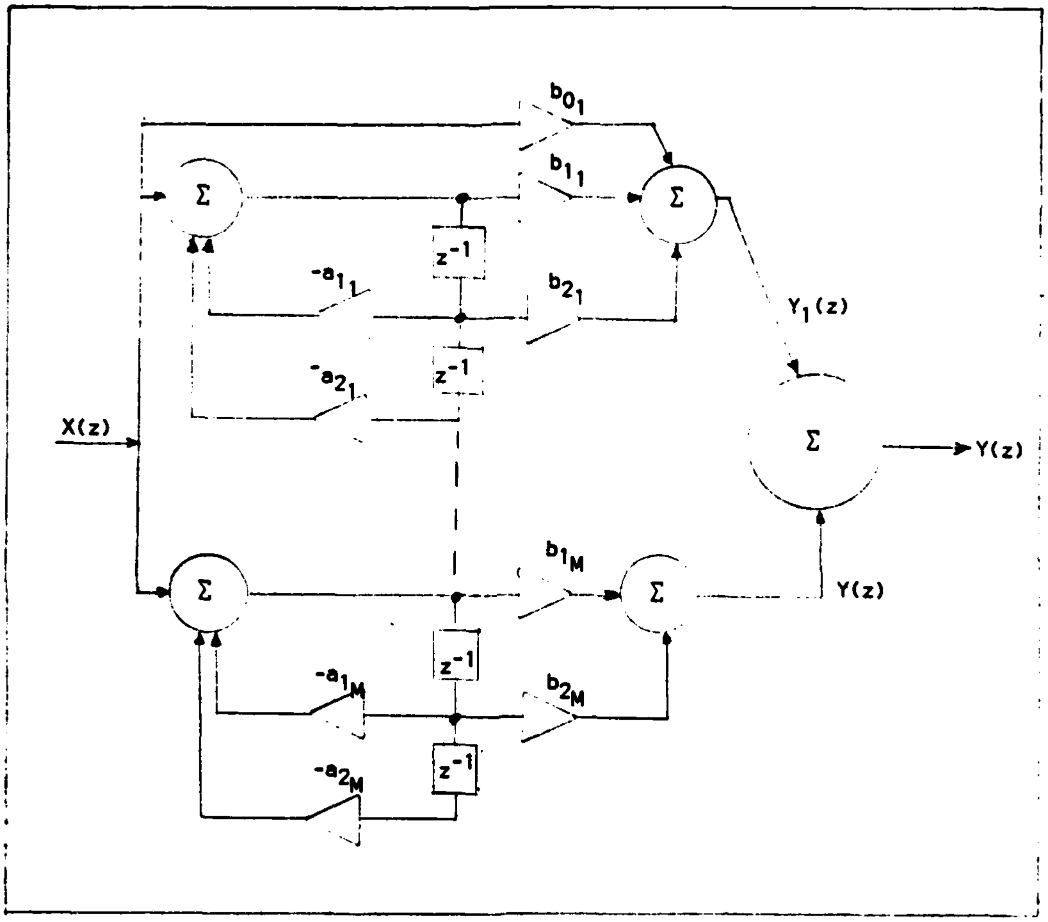


Figure 7. Parallel Form for IIR Digital Filters

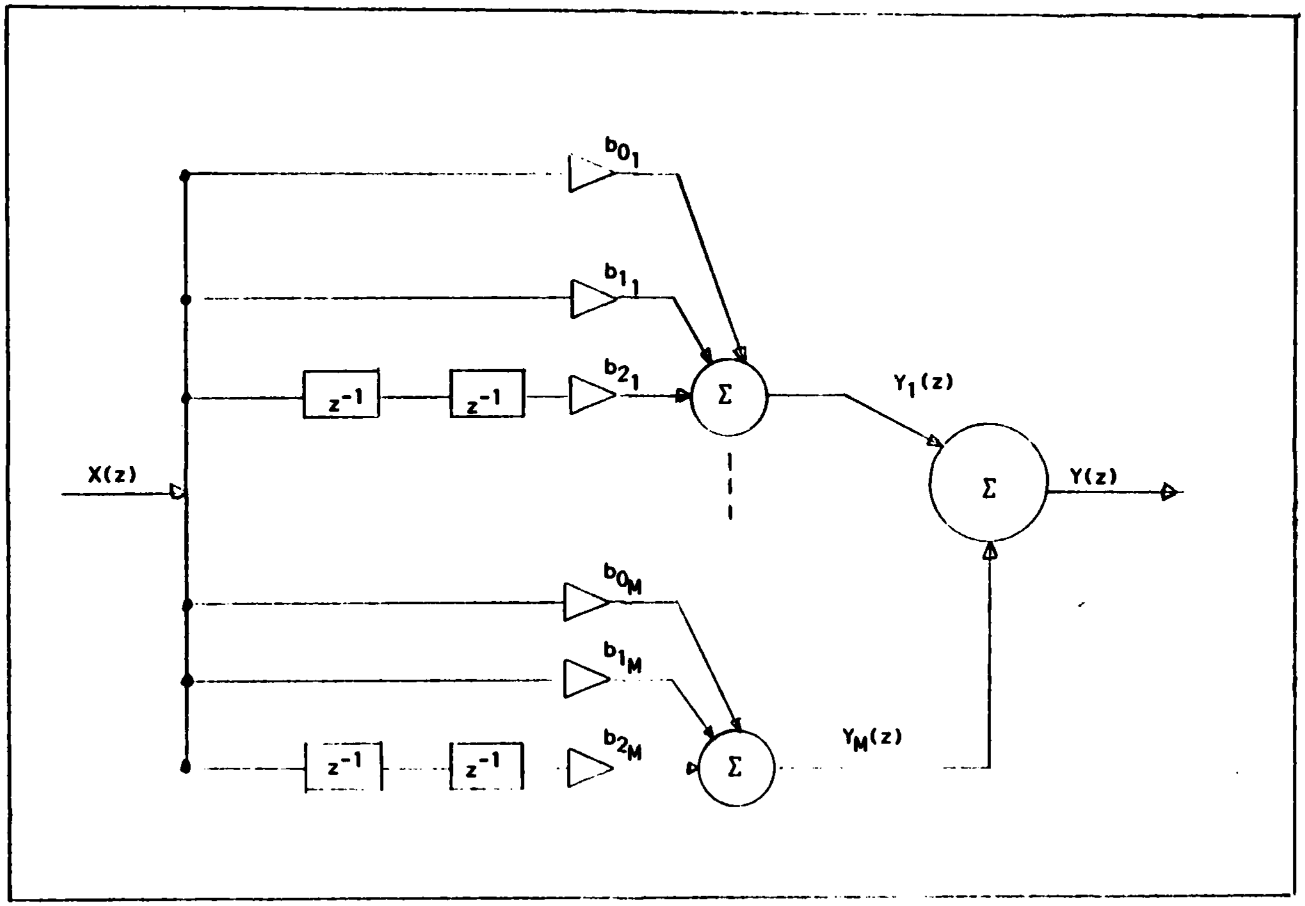


Figure 8. Parallel Form for FIR Digital Filters

coefficients is large (as is typically the case). The cascade form, on the other hand, reduces the dynamic range and thereby decreases sensitivity. But the realization in the latter case is more complicated because care must be taken to properly order various sections to avoid overflow and to minimize roundoff noise.

Nested structure promises to be an easy and attractive solution to the finite word length problems [13]. The transfer function of a nested structure filter can be easily derived by the nesting of the direct form transfer function H(z) as shown below.

IIR Filters. Instead of writing the summation in natural form, as shown in Equation (2-22), let it be arbitrarily permuted. Thus

$$H(z) = \frac{\sum_{k=0}^{M} b_{p_k} Z^{-p_k}}{1 + \sum_{k=1}^{N} a_{p_k} Z^{-p_k}} \qquad (3\text{-}10)$$

where p_k 's are the permuted elements of the set {0,1,2,...}. Equation (3-10) can be rewritten in the form

$$H(z) = \frac{b_{p_0} z^{-p_0} + b_{p_1} z^{-p_1} + \ldots + b_{p_M} z^{-p_M}}{1 + a_{p_1} z^{-p_1} + \ldots + a_{p_N} z^{-p_N}}$$

$$= \frac{c_0(z^{-p_0} + c_1(z^{-p_1} + \ldots + c_M z^{-p_M})\ldots)}{1 + d_1(z^{-p_1} + d_2(z^{-p_2} + \ldots + d_N z^{-p_N})\ldots)} \qquad (3\text{-}11)$$

where

$$c_0 = b_{p_0}$$

$$c_k = \frac{b_{p_k}}{b_{p_{k-1}}} \quad , \ k = 1,\ldots,M$$

$$d_1 = a_{p_1}$$

$$d_k = \frac{a_{p_k}}{a_{p_{k-1}}} \quad , \ k = 2,\ldots,N \tag{3-12}$$

Equation (3-11) can be written in alternative form

$$H(z) = \frac{c_0 z^{-p_0}(1 + c_1 z^{-p_1}(1 + \ldots + c_M z^{-p_M})\ldots)}{1 + d_1 z^{-p_1}(1 + d_2 z^{-p_2}(1 + \ldots + d_N z^{-p_N})\ldots)} \tag{3-13}$$

Corresponding filter structure for Equation (3-11) is shown in Figure 9 for the case $M = N$.

FIR Filters. Similar to the IIR case, Equation (2-25) can be permuted to obtain:

$$H(z) = \sum_{k=0}^{M} b_{p_k} z^{-p_k} \tag{3-14}$$

Equation (3-10) can be rewritten in an alternative form:

$$H(z) = e_0(z^{-p_0} + e_1(z^{-p_1} + \ldots + e_M z^{-p_M})\ldots) \tag{3-15}$$

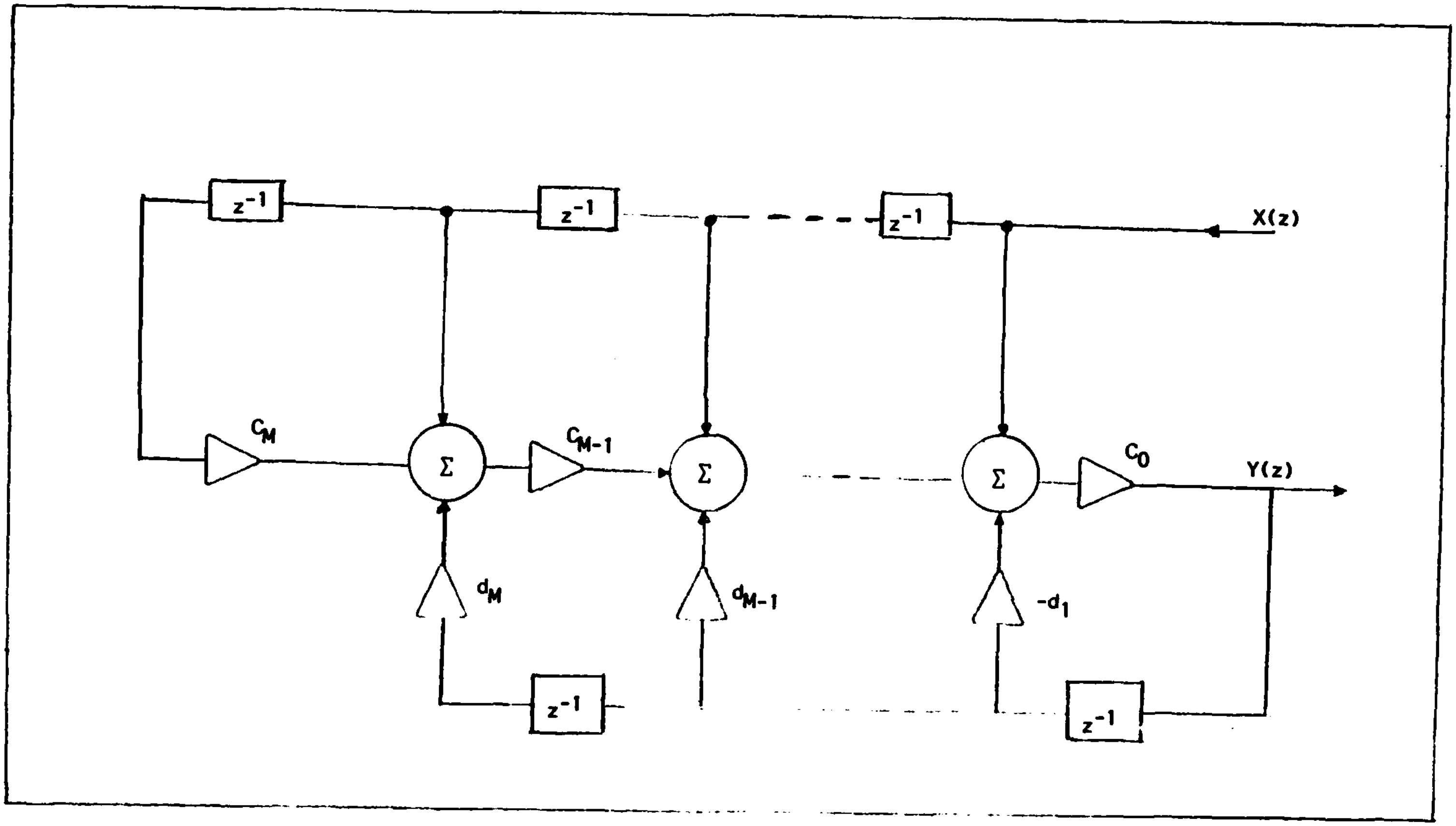


Figure 9. Nested Form for IIR Digital Filters

where

$$e_0 = b_{p_0}$$

$$e_n = \frac{b_{p_n}}{b_{p_{n-1}}} \quad , \quad n = 1 \text{ to } M \qquad (3\text{-}16)$$

Equation (3-15) can be expressed in a slightly different form as follows:

$$H(z) = e_0 z^{-p_0}(1 + e_1 z^{-p_1}(1 + \ldots + e_M z^{-p_M})\ldots) \qquad (3\text{-}17)$$

Corresponding filter structure for Equation (3-15) is shown in Figure 10.

Cascade-Nested Form

Similar to the direct form, the equation for a nested structure transfer function can be factored into numerous second order factors involving the powers z^{-2}, z^{-1}, and z^0. Each one of these second order polynomials is then realized as a second order filter section. Cascading these sections result in the required digital filter.

IIR Filters. Nested filter transfer function H(z), expressed by Equation (3-11) can be factored into a product of second order transfer functions as

$$H(z) = \prod_{i=1}^{M} H_i(z)$$

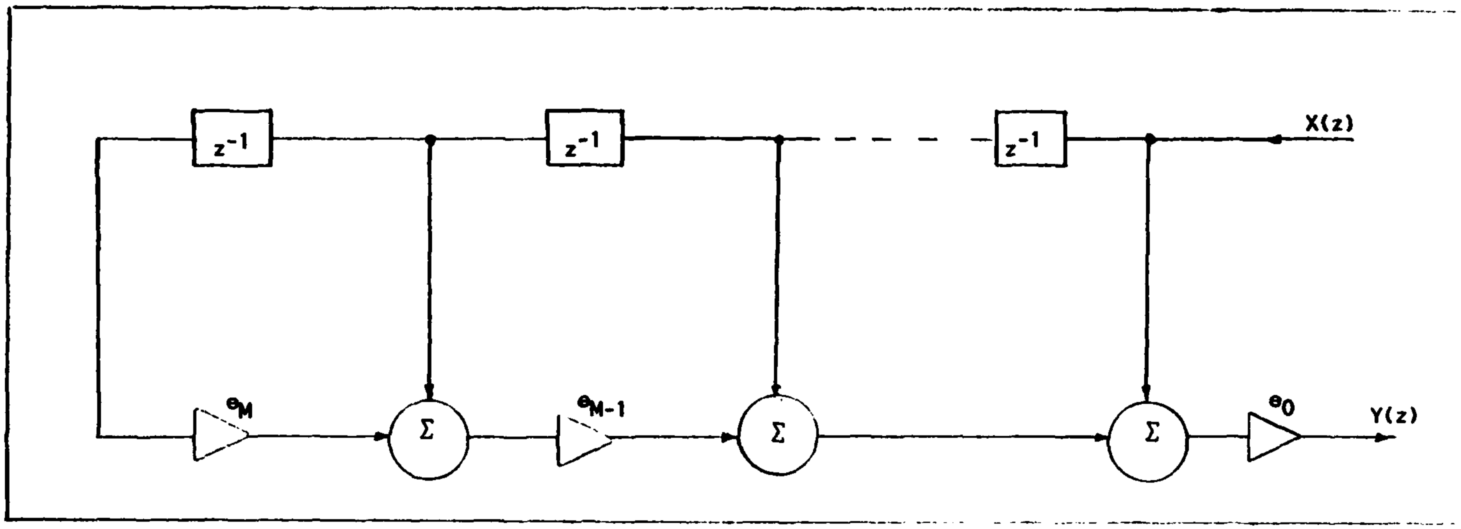


Figure 10. Nested Structure for FIR Digital Filter

where

$$H_i(z) = \frac{c_{0_i}(z^{-p_0} + c_{1_i}(z^{-p_1} + c_{2_i}z^{-p_2}))}{1 + d_{1_i}(z^{-p_1} + d_{2_i}z^{-p_2})} \qquad (3\text{-}18)$$

Corresponding filter structure is shown in Figure 11.

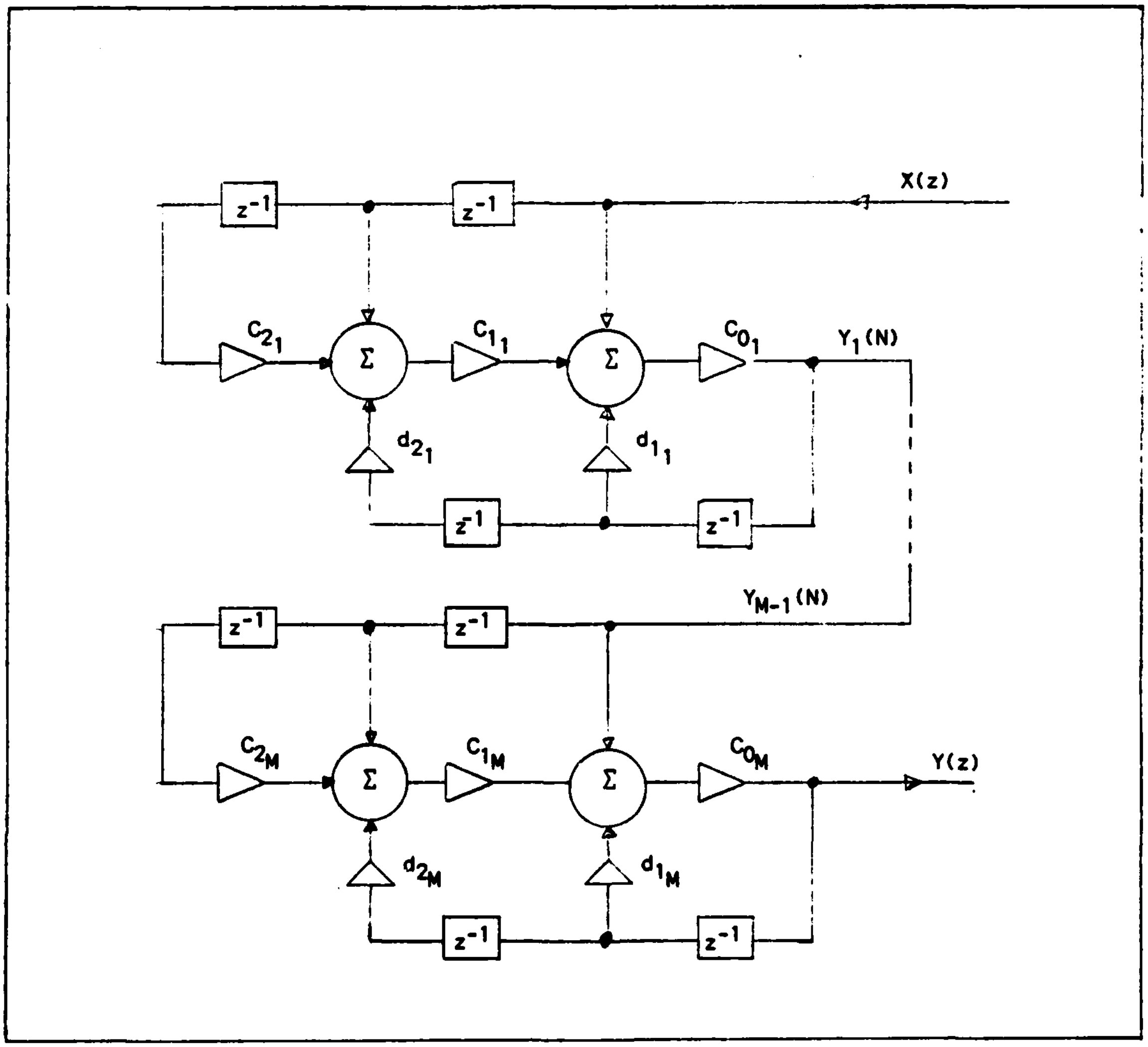


Figure 11. Cascade-Nested Structure for IIR Filters

FIR Filters. Similar to IIR filters, nested filter transfer function H(z) expressed by Equation (3-15) can be factored into a product of second order transfer functions as

$$H(z) = \prod_{i=1}^{M} H_i(z)$$

where

$$H_i(z) = e_0(z^{-p_0} + e_1(z^{-p_1} + e_2 z^{-p_2}))$$

Corresponding filter structure is shown in Figure 12.

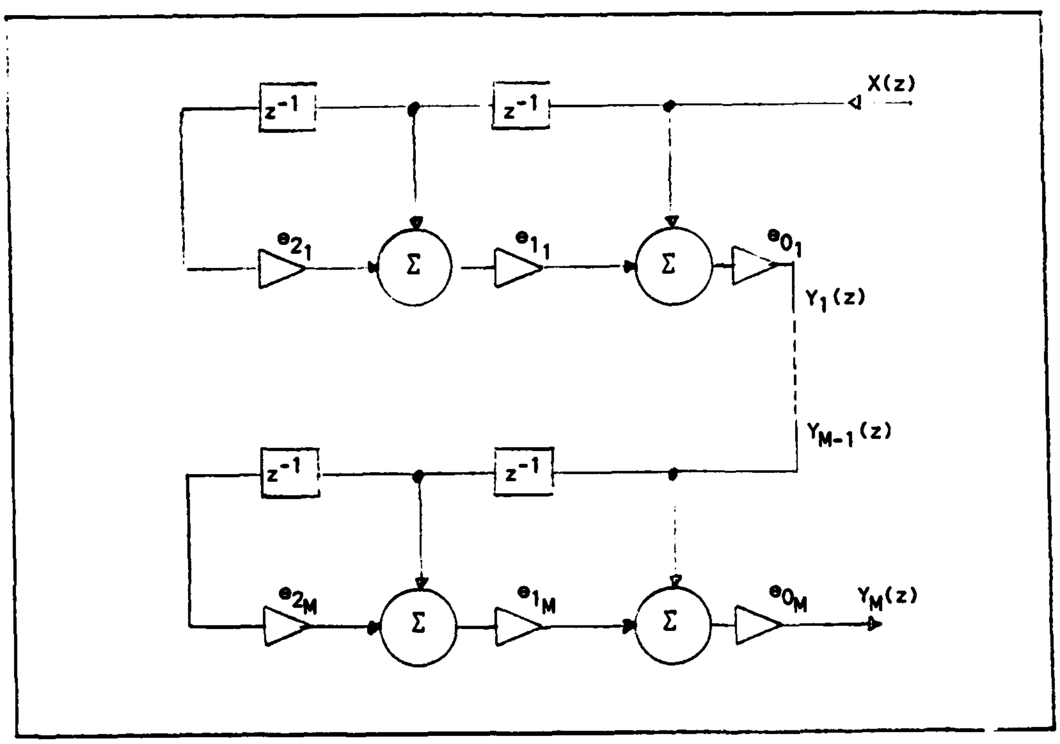


Figure 12. Cascade-Nested Form for FIR Digital Filters

Parallel-Nested Form

Similar to cascade form, parallel form is obtained by expanding the nested structure transfer function equations into numerous second order factors involving the power z^{-2}, z^{-1}, and z^{0}. Each one of these second-order factors is then realized as a second order filter section. Instead of cascading, as above, connecting these sections in parallel results in the required digital filter.

IIR Filters. The nested filter transfer function H(z) given by Equation (3-11) can be expressed as a partial fraction expansion in the form

$$H(z) = \sum_{i=1}^{M} H_i(z)$$

where $H_i(z)$ is the same as Equation (3-18).

Corresponding filter structure is shown in Figure 13.

FIR Filters. Similar to IIR filters, nested filter transfer function H(z) expressed by Equation (3-15) can be expressed as a partial fraction expansion in the form

$$H(z) = \sum_{i=1}^{M} H_i(z)$$

where $H_i(z)$ is the same as Equation (3-18).

Corresponding filter structure is shown in Figure 14.

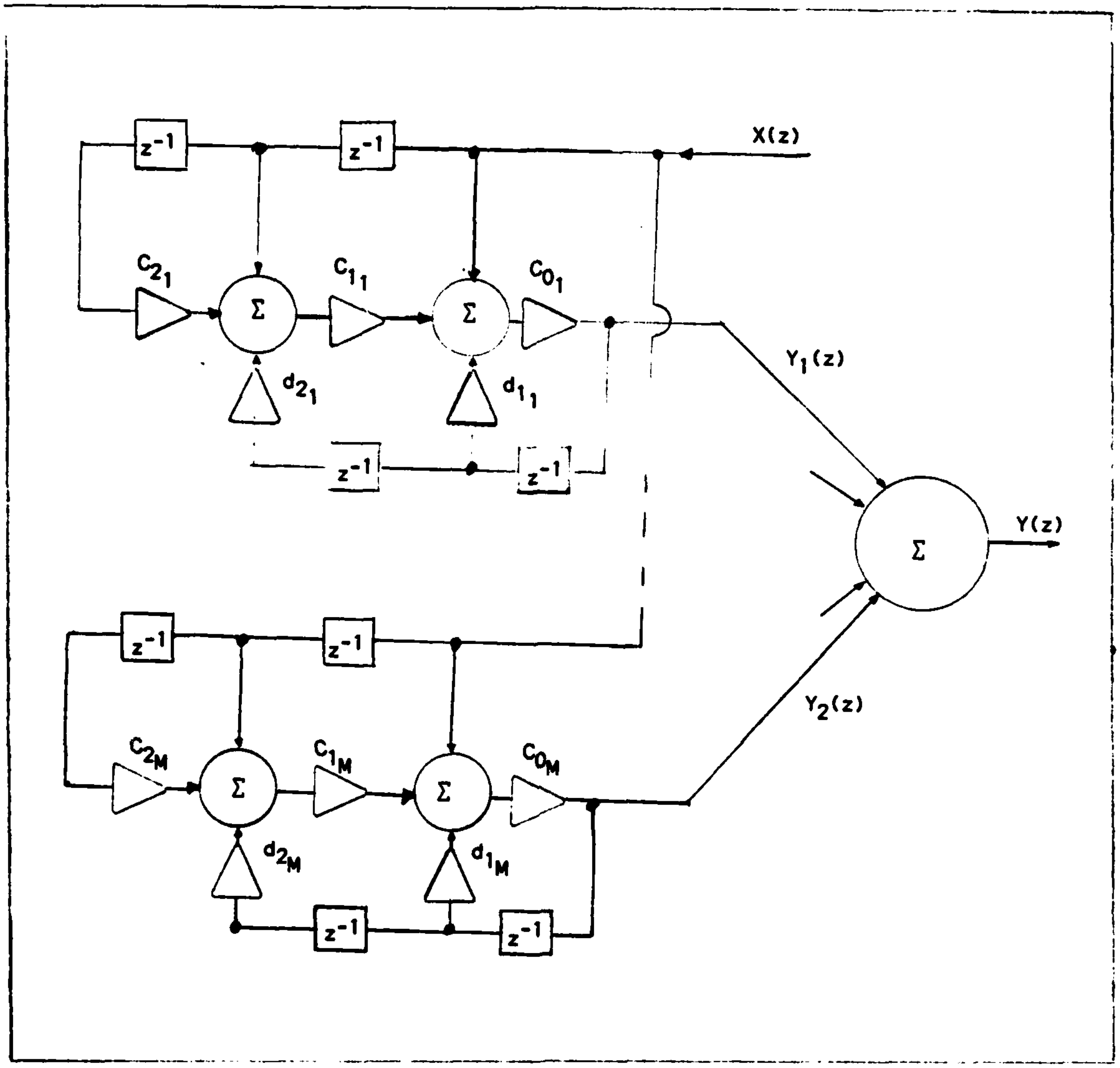


Figure 13. Parallel-Nested Structure for IIR Digital Filters

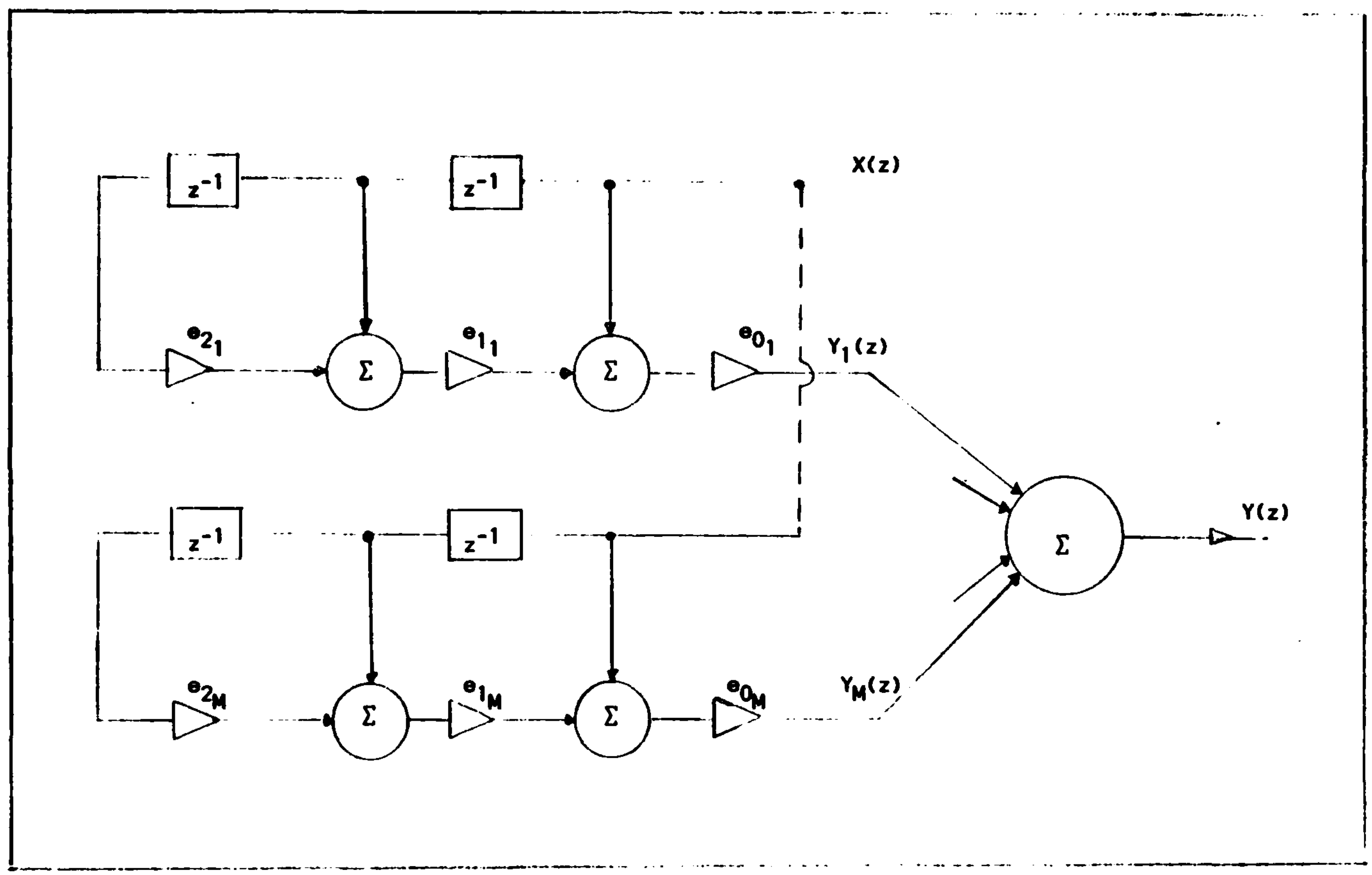


Figure 14. Parallel-Nested Structure for FIR Digital Filters

Sensitivity Analysis

The sensitivity is commonly defined as "Any change in the component characteristic that causes a change in the transfer function." In digital filter implementation, the desired transfer function is calculated on the basis of infinite precision arithmetic. But, in actuality, all the components, like multipliers, storage devices, and adders, work with finite number of bits. This fact will cause the change in the transfer function of the digital filter which is calculated based on infinite precision. This change is known as the sensitivity of the transfer function and is given by:

$$S_{\alpha_i}\{|H(z)|\} = \mathrm{Re}\{S_{\alpha_i} H(z)\} = \mathrm{Re}\{\frac{\alpha_i}{H(z)} \frac{\partial H(z)}{\partial \alpha_i}\} \qquad (3\text{-}19)$$

where H(z) is the transfer function of the digital filter, and α_i is the system parameter that varies. There are many different criteria of sensitivity that have been used in digital filter implementation. However, the fractional change in the transfer function magnitude due to a change in the multiplier coefficients, or the change in the location of the poles due to change in the multiplier coefficients are, in most cases, reasonable criteria of sensitivity.

As we pointed out earlier in this chapter, different filter structures for the same transfer function have different response characteristics. In other words, sensitivity of a digital filter depends heavily upon the particular

realization. We next examine the sensitivity versus realization relationship for the various realizations discussed so far in this thesis.

Sensitivity Analysis in IIR Filters

Direct Form. Let us rewrite Equation (2-22) as:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

The multiplier coefficient a_k and b_k will be quantized to $\hat{a}_k$ and $\hat{b}_k$. Thus,

$$\begin{aligned} \hat{a}_k &= a_k - \Delta a_k \\ \hat{b}_k &= b_k - \Delta b_k \end{aligned} \qquad (3\text{-}20)$$

where Δa_k and Δb_k are error quantities which are statistically independent and uniformly distributed [10]. Therefore, the realized transfer function will be

$$\hat{H}(z) = \frac{\sum_{k=0}^{M} \hat{b}_k z^{-k}}{1 + \sum_{k=1}^{N} \hat{a}_k z^{-k}} \qquad (3\text{-}21)$$

If we let $\hat{y}(n)$ denote the actual filter output and let $y(n)$ denote the ideal filter output due to the same input $x(n)$, then by using Equation (2-20) the error $e(n)$ in the

two outputs is given by

$$e(n) = \hat{y}(n) - y(n) \tag{3-22}$$

or

$$\begin{aligned} e(n) = & \sum_{k=0}^{M} \Delta b_k x(n-k) - \sum_{k=1}^{N} a_k e(n-k) \\ & - \sum_{k=1}^{N} \Delta a_k y(n-k) - \sum_{k=1}^{N} \Delta a_k e(n-k) \end{aligned} \tag{3-23}$$

Assuming that the error $e(\cdot)$ and the quantization errors Δa_k are small, the last term in Equation (3-23) can be neglected. Furthermore, if we let M equal to N, Equation (3-23) can be written as

$$e(n) = \sum_{k=0}^{N} \Delta b_k x(n-k) - \sum_{k=1}^{N} a_k e(n-k) - \sum_{k=1}^{N} \Delta a_k y(n-k) \tag{3-24}$$

Combining and taking the Z-transform of Equation (3-22) and Equation (3-24) will give

$$\begin{aligned} \hat{y}(z) - y(z) = & \sum_{k=0}^{N} \Delta b_k z^{-k} x(z) - \sum_{k=1}^{N} a_k z^{-k} \\ & \cdot (\hat{y}(z) - y(z)) - \sum_{k=1}^{N} \Delta a_k z^{-k} y(z) \end{aligned} \tag{3-25}$$

If we substitute $y(z) = H(z)X(z)$ and $\hat{y}(z) = \hat{H}(z)X(z)$ into Equation (3-25), the resulting equation can be arranged as

$$\hat{H}(z) - H(z) = \frac{\sum_{k=0}^{N} \Delta b_k z^{-k} - \sum_{k=1}^{N} \Delta a_k z^{-k} H(z)}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{3-26}$$

Here $\hat{H}(z) - H(z)$ is a measure of the deviation of the frequency response of the actual filter from the frequency response of the ideal filter. In filter implementation, one possible measure of the effect of coefficient quantization is the mean-square error in the frequency response, and can be defined in terms of $H(\cdot)$ and $\hat{H}(\cdot)$ as

$$\sigma_{\Delta_H}^{2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} |\hat{H}(e^{j\omega}) - H(e^{j\omega})|^{2} d \tag{3-27}$$

where $\hat{H}(e^{j\omega})$ and $H(e^{j\omega})$ denote the quantized and ideal frequency response of the transfer function, respectively. Using the assumed statistical independence among Δb_k and Δa_k, and substituting Equation (3-26) into (3-27), the last equation reduces to

$$\sigma_{\Delta_H}^{2} = \sum_{k=0}^{N} \Delta b_k^{2} \; \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{\left(1+ \sum_{k=1}^{N} a_k z^{-k}\right)\left(1+ \sum_{k=1}^{N} a_k z^{k}\right)} \frac{dz}{z}$$

$$+ \sum_{k=1}^{N} \Delta a_k^{2} \; \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\left(\sum_{k=0}^{N} b_k z^{-k}\right)\left(\sum_{k=0}^{N} b_k z^{k}\right)}{\left(1+ \sum_{k=1}^{N} a_k z^{-k}\right)^{2}\left(1+ \sum_{k=1}^{N} a_k z^{k}\right)^{2}} \frac{dz}{z} \tag{3-28}$$

Equation (3-28) may be evaluated to any degree of accuracy using a short digital computer program based on Figure 15.

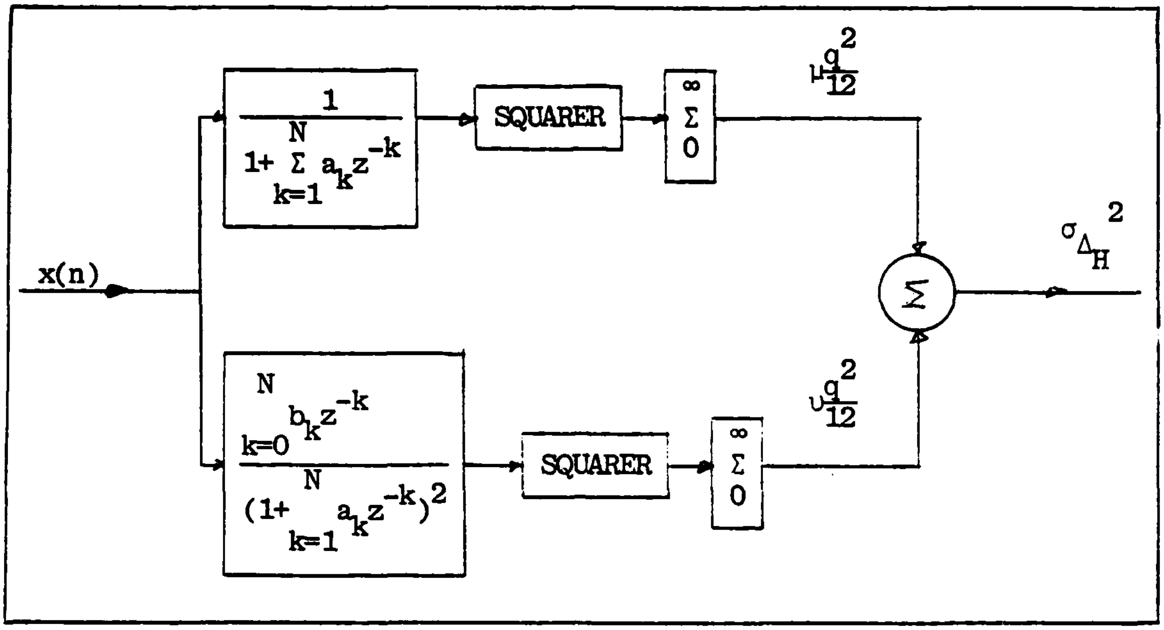


Figure 15. Technique for Measuring Variance of Error Due to Coefficient Quantization

If the quantization is carried out by rounding with quantization in steps of q, then Δb_k and Δa_k can assume any value at random in the range $-\frac{q}{2}$ to $+\frac{q}{2}$; that is, Δb_k and Δa_k are uniformally distributed between $-\frac{q}{2}$ to $+\frac{q}{2}$. The quantization step q is equal to 2^{-t} , where t is the number of bit used in the register to store the number. Since the probability density $p(\cdot)$ of Δa_k or Δb_k is assumed to be uniform, we have

$$p(\Delta a_k) = p(\Delta b_k) = \begin{cases} \frac{1}{q} & \text{for } -\frac{q}{2} \le \Delta a_k \le \frac{q}{2} \\ 0 & \text{otherwise.} \end{cases} \qquad (3\text{-}29)$$

Therefore, the mean and the variance of Δa_k as well as Δb_k are given by

$$E[\Delta a_k] = E[\Delta b_k] = 0 \tag{3-30}$$

$$\sigma_{\Delta a_k}^{2} = \sigma_{\Delta b_k}^{2} = \frac{q^2}{12} \tag{3-31}$$

Substituting Equation (3-31) into Equation (3-28), and denoting by $\sigma_{\Delta H_D}$ the error variance for the direct form realization, we get

$$\sigma_{\Delta H_D}^{2} = \left(\sum_{k=0}^{N} \sigma_{\Delta b_k}\right)^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{\left(1+\sum_{k=1}^{N} a_k z^{-k}\right)\left(1+\sum_{k=1}^{N} a_k z^{k}\right)} \frac{dz}{z}$$

$$+\left(\sum_{k=1}^{N} \sigma_{\Delta a_k}\right)^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\left(\sum_{k=0}^{N} b_k z^{-k}\right)\left(\sum_{k=0}^{N} b_k z^{k}\right)}{\left(1+\sum_{k=1}^{N} a_k z^{-k}\right)^2\left(1+\sum_{k=1}^{N} a_k z^{k}\right)} \frac{dz}{z} \tag{3-32}$$

where

$$\sum_{k=0}^{N} \sigma_{\Delta a_k}^{2} = \mu \frac{q^2}{12}$$

$$\sum_{k=1}^{N} \sigma_{\Delta b_k}^{2} = \upsilon \frac{q^2}{12} \tag{3-33}$$

and μ and υ are the number of nonzero coefficients in the numerator and denominator of Equation (3-1), respectively.

Kaiser was one of the first to investigate the effect of coefficient errors [11] on filter performance. Kaiser

pointed out that small errors in the coefficients can cause large shifts in the poles (or zeros) of the direct form narrow-band IIR digital filters [11]. To see this, let us suppose that the poles of $H(z)$ are located at $z=z_i$, $i=1,2,\ldots,N$ and that the poles of $\hat{H}(z)$ are located at $z=z_i+\Delta z_i$, $i=1,2,\ldots,N$. Furthermore, let us rewrite the denominator of Equation (2-22) in factored form as

$$p(z) = 1 - \sum_{k=1}^{N} a_k z^{-k} = \prod_{k=1}^{N} (1 - a_k z^{-1}) \qquad (3\text{-}34)$$

The error Δz_i can be expressed in terms of the errors in the coefficient as

$$\Delta z_i = \sum_{k=1}^{N} \frac{\partial z_i}{\partial a_k} \Delta a_k \qquad i=1,2,\ldots,N \qquad (3\text{-}35)$$

Using Equation (3-34):

$$\left(\frac{\partial p(z)}{\partial z_i}\right)_{z=z_i} \cdot \frac{\partial z_i}{\partial a_k} = \left(\frac{\partial (p(z))}{\partial a_k}\right)_{z=z_i}$$

$$\frac{\partial z_i}{\partial a_k} = \frac{z_i^{N-k}}{\prod\limits_{\substack{\ell=1 \\ \ell\neq 1}}^{N} (z_i = z\ell)} \qquad (3\text{-}35)$$

The poles of some $H(z)$ are shown in Figure 16 for discussion. The magnitude of the denominator of Equation (3-36) is equal to the product of the lengths of the vectors from all the remaining poles to the pole z_i shown in Figure 16.

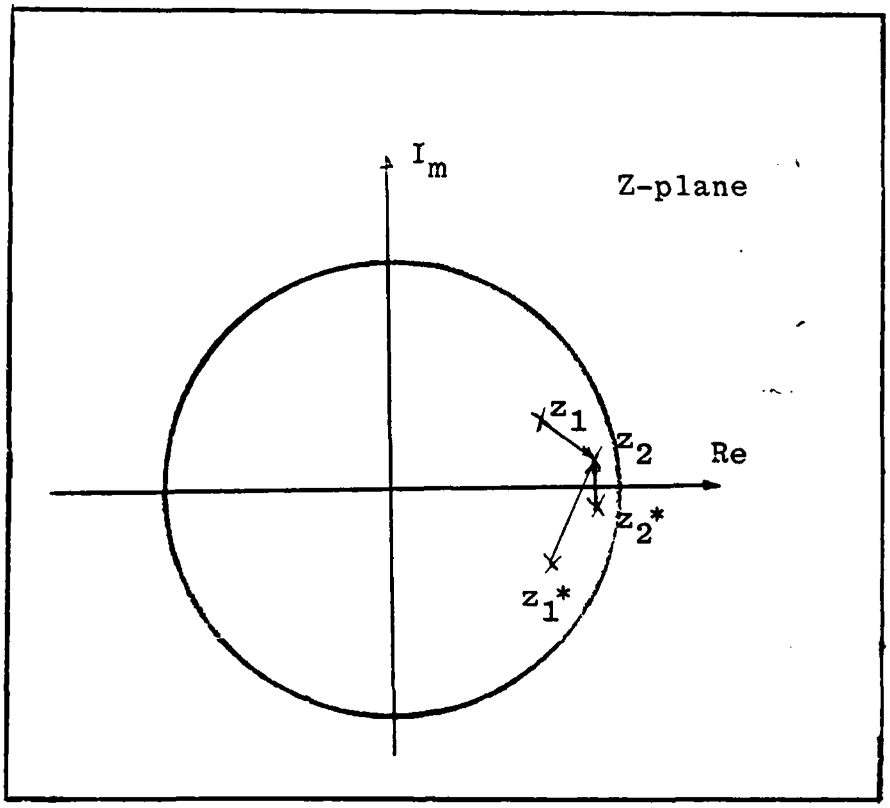


Figure 16. Representation of the Factors of Equation (3-40) as vectors in Z-Plane

If the poles are very close to each other, then small changes in coefficients will cause relatively large changes in the location of poles. In other words, system will be too sensitive to coefficient change. Furthermore, it is evident that the larger the number of roots, the greater is the sensitivity.

Cascade Form. The actual transfer function of digital filter realized in cascade form can be expressed as

$$\hat{H}(z) = \prod_{i=1}^{N} \hat{H}_i(z) \tag{3-37}$$

where

$$\hat{H}_i(z) = \frac{\hat{b}_{0i} + \hat{b}_{1i}z^{-1} + \hat{b}_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}} \tag{3-38}$$

and N equals number of second order section. Each second-order section contributes an uncorrelated error component as described by Equation (3-32), and the total output error is obtained by summing these various errors weighted by the transfer function from their point of injection to their respective outputs.

The output mean-squared error $\sigma_{\Delta H_C}{}^2$ can be easily computed as follows by using the error model given in Figure 17.

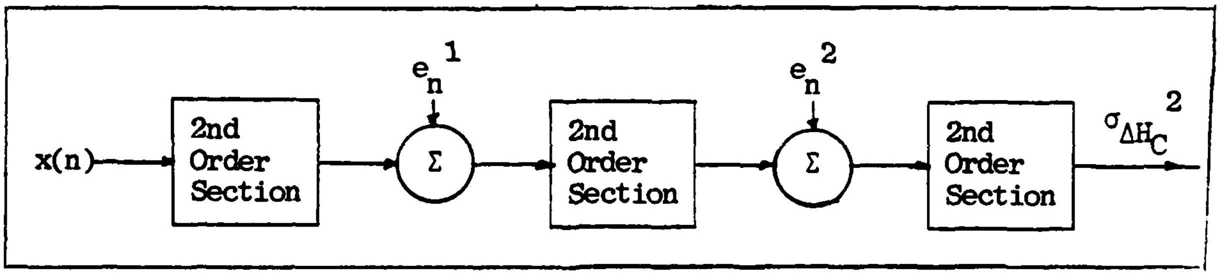


Figure 17. Error Model for Cascade Form

$$\sigma_{\Delta H_C}{}^2 = \sum_{j=1}^{N-1} \frac{\sigma^j_{\Delta H_D}{}^2}{2\pi i} \int_{-\pi}^{\pi} H^j(z)H^j(z^{-1}) \frac{dz}{z} \tag{3-39}$$

where $\sigma^j_{\Delta H_D}{}^2$ can be found from Equation (3-31) by letting $N=2$, $H^j(z)$ is the transfer function between the output of the jth second order section and its input. Comparison of

$\sigma_{\Delta H_C}$ with $\sigma_{\Delta H_D}$ is made by Knowles and Olcayto [12].

Since each pair of complex-conjugate poles is realized separately, the error in a given pole is independent of its distance from the other poles of the system. For this reason, cascade form is to be preferred over the direct form in the implementation of narrow-band IIR digital filter.

Parallel Form. The actual transfer function of digital filter formed in parallel can be expressed as

$$\hat{H}(z) = \sum_{i=1}^{N} \hat{H}_i(z) \tag{3-40}$$

where $\hat{H}_i(z)$ and N are the same as in the Equation (3-37).

As we expressed in cascade case, each second-order section contributes an uncorrelated error component as described in Equation (3-30) and the output error is simply the sum of the various errors from the second order sections. In Figure 18, the error model is shown for the parallel form.

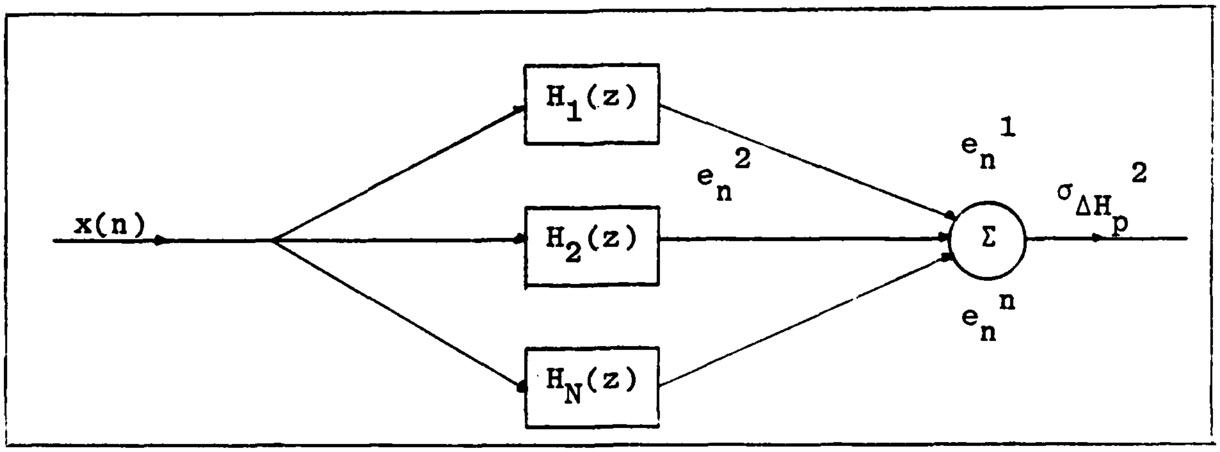


Figure 18. Error Model for Parallel Form

The output mean squared error can be easily computed using the error model given in Figure 18.

$$\sigma_{\Delta H_P}{}^2 = \sum_{j=1}^{N} (\sigma_{\Delta H_D}{}^j)^2 \qquad (3\text{-}41)$$

where $\sigma_{\Delta H_P}$ is the error in parallel form realization and N is the number of second order sections. Parallel form is to be preferred over the direct form in the implementation of narrow-band IIR digital filter because of the same reasons given for the cascade form.

Nested Structure. The nested structure transfer function was derived in the last section. Let us rewrite it below for convenience.

$$H(z) = \frac{C_0(z^{-p_0} + c_1(z^{-p_1} + \ldots + c_M z^{-p_M}) \ldots)}{1 + d_1(z^{-p_1} + d_2(z^{-p_2} + \ldots + d_N z^{-p_N}) \ldots)}$$

where

$$c_0 = b_0$$

$$c_k = \frac{b_{p_k}}{b_{p_{k-1}}} \quad , \; k=1,2,\ldots,M$$

$$d_1 = a_1$$

$$d_k = \frac{a_{p_k}}{a_{p_{k-1}}} \qquad , \; k=2,3,\ldots,N$$

so that

$$b_{p_k} = \prod_{n=0}^{k} c_n \qquad k=1,2,\ldots,M$$

$$a_{p_k} = \prod_{n=0}^{k} d_n \qquad k=2,3,\ldots,N \qquad (3\text{-}42)$$

When the nested structure filter coefficients c_k and d_k are rounded, the realized filter will have an effective b_{p_k}'s and a_{p_k}'s given by

$$\hat{b}_{p_k} = \prod_{n=0}^{k} (c_n)r \quad , \; k=1,2,\ldots,M$$

$$\hat{a}_{p_k} = \prod_{n=1}^{k} (d_n)r \quad , \; k=2,3,\ldots,N \qquad (3\text{-}43)$$

where "^" denotes the effective value, and the subscript r denotes the rounding operation.

The relative errors b_{p_k} and a_{p_k}, given by E_k/b_{p_k} and E_k/a_{p_k} respectively, tend to grow with k, due to the cumulative errors in c_0 through c_k and in d_1 through d_k. Therefore, we redefine c_k's and d_k's as

$$c_0 = b_0$$

$$c_k = \frac{b_{p_k}}{\hat{b}_{p_{k-1}}} = \frac{b_{p_k}}{\prod_{n=0}^{k-1} (c_n)_r} \quad , \quad k=1,2,\ldots,M$$

$$d_1 = a_1$$

$$d_k = \frac{a_{p_k}}{\hat{a}_{p_{k-1}}} = \frac{a_{p_k}}{\prod_{n=0}^{k-1} (d_n)_r} \quad , \quad k=2,3,\ldots,N \tag{3-44}$$

Now, the effective b_{p_k} becomes

$$\hat{b}_{p_k} = \prod_{n=0}^{k-1} (c_n)_r \quad (c_k)_r \tag{3-45}$$

where

$$(c_k)_r = c_k + \varepsilon_k \qquad k=1,2,\ldots,M \tag{3-46}$$

where ε_k is the rounding error. By combining Equations (3-46) and (3-44) and substituting into Equation (3-45), we get

$$\hat{b}_{p_k} = \hat{b}_{p_{k-1}} \left[\frac{b_{p_k}}{b_{p_{k-1}}} + k \right]$$

$$\hat{b}_{p_k} = b_{p_k} + \hat{b}_{p_{k-1}} \varepsilon_k \qquad k=1,2,\ldots,M \tag{3-47}$$

Therefore, the error in coefficients b_{p_k}, will be

$$E_{b_k} = \hat{b}_{p_k} - b_{p_k}$$

$$E_{b_k} = \hat{b}_{p_{k-1}} \varepsilon_k \quad , \quad k=1,2,\ldots,M \tag{3-48}$$

Similarly, the error in coefficients a_{p_k} can be derived, with the result

$$E_{a_k} = \hat{a}_{p_{k-1}} \varepsilon_k \quad , \quad k=2,3,\ldots,N \tag{3-49}$$

The mean-square error in the frequency response can be derived from Equation (3-28). In Equation (3-48), E_{b_k}, and in Equation (3-49), E_{a_k}, are equal to Δb_k and Δa_k, respectively. If we substitute Equations (3-48) and (3-49) into Equation (3-28) and assume that M equals N for simplicity, then the mean square error $\sigma_{\Delta H_{ND}}{}^2$ for the direct form nested structure will become

$$\sigma_{\Delta H_{ND}}{}^2 = \left[\sum_{k=0}^{N} (\hat{b}_{p_{k-1}} \varepsilon_k)^2\right] \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{(1+\sum_{k=1}^{N} a_k z^{-k})(1+\sum_{k=1}^{N} a_k z^{k})} \frac{dz}{z}$$

$$+ \left[\sum_{k=1}^{N} (\hat{a}_{p_{k-1}} \varepsilon_k)^2 \frac{1}{2\pi}\right] \int_{-\pi}^{\pi} \frac{\left(\sum_{k=0}^{N} b_k z^{-k}\right)\left(\sum_{k=0}^{N} b_k z^{k}\right)}{\left(1+\sum_{k=1}^{N} a_k z^{-k}\right)\left(1+\sum_{k=1}^{N} a_k z^{k}\right)} \frac{dz}{z} \tag{3-50}$$

Similarly, cascade and parallel form nested structure can be derived using the above procedure. The resulting mean square error $\sigma_{\Delta H_{NC}}{}^2$ for cascade nested structure will be

$$\sigma_{\Delta H_{NC}}{}^2 = \sum_{j=1}^{N-1} \frac{(\sigma_{\Delta H_{ND}}^{\,j})^2}{2\pi} \int_{-\pi}^{\pi} H^j(z)H^j(z^{-1})\,\frac{dz}{z} \tag{3-51}$$

where $(\sigma_{\Delta H_{ND}}^{\,j})^2$ can be found from Equation (3-50) by letting N=2. $H^j(z)$ is the transfer function between the output of the jth second order section and its input. The same error model for computation can be used as shown in Figure 17.

Similarly, the result for parallel-nested structure will be

$$\sigma_{\Delta H_{NP}}{}^2 = \sum_{j=1}^{N} (\sigma_{\Delta H_{ND}}^{\,j})^2 \tag{3-52}$$

where $\sigma_{H_{NP}}$ is the error in parallel form realization. Figure 18 can be used for error model.

Sensitivity Analysis in FIR Filters

Direct Form. The transfer function of an FIR filter is given in Equation (3-3). Let us rewrite the above equation for convenience below.

$$H(z) = \sum_{k=0}^{M} h(k)z^{-k}$$

After h_k 's are quantized, the realized transfer function will be

$$\hat{H}(z) = \sum_{k=0}^{M} \hat{h}(k) z^{-k} \qquad (3\text{-}53)$$

As before, the measure of the effect of coefficient quantization is the error in the frequency response which is denoted as

$$|E(e^{j\omega})|_D = |\hat{H}(e^{j\omega}) - H(e^{j\omega})| \qquad (3\text{-}54)$$

Therefore,

$$|E(e^{j\omega})|_D = \sum_{k=0}^{M} \Delta h(k) \qquad (3\text{-}55)$$

Since $|\Delta h(k)| \leq q/2$, where q is quantization step,

$$|E(e^{j\omega})|_D \leq N\ q/2 \qquad (3\text{-}56)$$

Cascade Form. The actual transfer function of digital filter formed in cascade can be expressed as

$$\hat{H}(z) = \prod_{i=1}^{N} \hat{H}_i(z) \qquad (3\text{-}57)$$

where $\hat{H}_i(z) = \hat{b}_{0_i} + \hat{b}_{1_i} z^{-1} + \hat{b}_{2_i} z^{-2}$ (3-58)

and N is the number of second order section.

Denoting by $|E(e^{j\omega})|_C$ the error in the frequency response of this filter due to quantization, we can write

$$|E(e^{j\omega})|_C = \sum_{i=1}^{N-1} |E^i(e^{j\omega})^i|_D \; |H^i(e^{j\omega})| \qquad (3\text{-}59)$$

where $E^i(e^{j\omega})$ can be found from Equation (3-55) by letting $M=2$; $H^i(e^{j\omega})$ in the above equation is the transfer function relating the output of the i^{th} second-order section to its input.

Parallel Form. The actual transfer function of digital filter implemented in the parallel form can be expressed as

$$\hat{H}(z) = \sum_{i=1}^{N} \hat{H}_i(z) \qquad (3\text{-}60)$$

where $\hat{H}_i(z)$ and N are the same as in Equation (3-58). The output error in frequency domain is simply the sum of the various errors from the second order sections. Thus, denoting by $|E(e^{j\omega})|_P$ the error in the frequency of this filter due to quantization, we get

$$|E(e^{j\omega})|_P = \sum_{i=1}^{N} |E^i(e^{j\omega})|_D \qquad (3\text{-}61)$$

where $|E^i(e^{j\omega})|_D$ is the same as in Equation (3-59).

Nested Structure. The nested structure filter transfer function was derived in the last section. Recall

that the transfer function was expressed in Equation (3-14) and the nested form transfer function was given in Equation (3-15). Let us rewrite these equations below for convenience.

$$H(z) = \sum_{k=0}^{M} b_{p_k} z^{-p_k} \qquad (3\text{-}62)$$

$$H(z) = e_0(z^{-p_0} + e_1(z^{-p_1} + \dots + e_M z^{-p_M}) \qquad (3\text{-}63)$$

where

$$e_o = b_{p_0}$$

$$e_n = \frac{b_{p_n}}{b_{p_{n-1}}}$$

so that

$$b_{p_n} = \prod_{k=0}^{n} e_k \qquad (3\text{-}64)$$

The relative error in b_{p_n} is given by $\frac{E_n}{b_{p_n}}$, where $E_n = \hat{b}_{p_n} - b_{p_n}$ tends to grow with n , due to cumulative errors in e_0 through e_n . Therefore, we redefine e_n 's as follows:

$$e_0 = b_{p_0}$$

$$e_n = \frac{b_{p_n}}{\hat{b}_{p_{n-1}}} = \frac{b_{p_n}}{\prod_{k=0}^{n-1} (e_n)_r} \qquad n=1,\ldots,n-1 \tag{3-65}$$

where "^" stands for effective value and "r" stands for rounding operation in Equation (3-65). Thus

$$(e_n)_r = e_n + \varepsilon_n \tag{3-66}$$

where ε_n is the rounding error and is the same as explained in IIR section. The effective b_{p_n} now becomes,

$$\hat{b}_{p_n} = \sum_{k=0}^{n-1} (e_n)_r \quad (e_n)_r = \hat{b}_{p_{n-1}} \frac{b_{p_n}}{\hat{b}_{p_{n-1}}} + \varepsilon_n \tag{3-67}$$

The error in coefficient b_{p_n} 's will be

$$E_n = \hat{b}_{p_n} - b_{p_n}$$

$$E_n = \hat{b}_{p_{n-1}} \varepsilon_n \tag{3-68}$$

Then the error quantity in frequency response can be computed as

$$E(e^{j\omega}) = \hat{H}(e^{j\omega}) - H(e^{j\omega})$$

$$|E(e^{j\omega})| = \sum_{n=0}^{M} |\hat{b}_{p_{n-1}} \varepsilon_n| \tag{3-69}$$

Summary

This chapter was directed toward the realization and the related cause of sensitivity of digital filters. A number of structures, such as direct, cascade, parallel, nested, cascade-nested, and parallel-nested, were presented for IIR and FIR filters.

One of the most important considerations in the choice of a structure (realization) for implementation of a filter is the low sensitivity. Thus, we presented the sensitivity analysis for the various structures mentioned above.

IV. Simulation of Digital Filters

Introduction

In this chapter we will simulate the FIR digital filters, discussed in previous chapters, using many different word lengths. The input-output relationship of the FIR digital filters are given by Equation (2-23). First, FIR digital filter coefficients and input in this equation will be obtained according to user requirements. The input, which is designed such that its values are all positive to handle the two's complement addition easily, can be step, multiple-step or sinusoidal function. Second, these coefficients and input will be scaled to prevent the overflow at the output of the digital filter. The absolute maximum value of the scaled input signal will be less than .1. Since the scaling technique for coefficients depends on the type of filter, it will be discussed in the Simulation I section. Third, all the numbers pertaining to the filters will be quantized according to user requirements by either truncation or rounding. Finally, the simulation results depicting filter outputs will be obtained based on these quantized data.

Simulation I

The FIR digital filters will be simulated based on 10 bits word length register. The input function to all

the digital filters for simulation I will be the same as shown in Figure 19. Corresponding input values for 20 points is shown in the first column of Table I. The quantized input is shown in the second column and the scaled version of the input appears in the third column of the same table.

TABLE I

INPUT SEQUENCES

$\underline{x}$	$\hat{\underline{x}}_s$	$\underline{x}_s$
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.1000000E 00	.9960938E-01	.1000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00

<u>Direct Form</u>. The fourth order low-pass FIR digital filter coefficients with the normalized cut-off frequency of .17 are obtained by using a rectangular weighting window. Then, these coefficients are scaled, such that the summation of the absolute value of the coefficients is less than .1, to prevent overflow. Finally, the scaled coefficients are

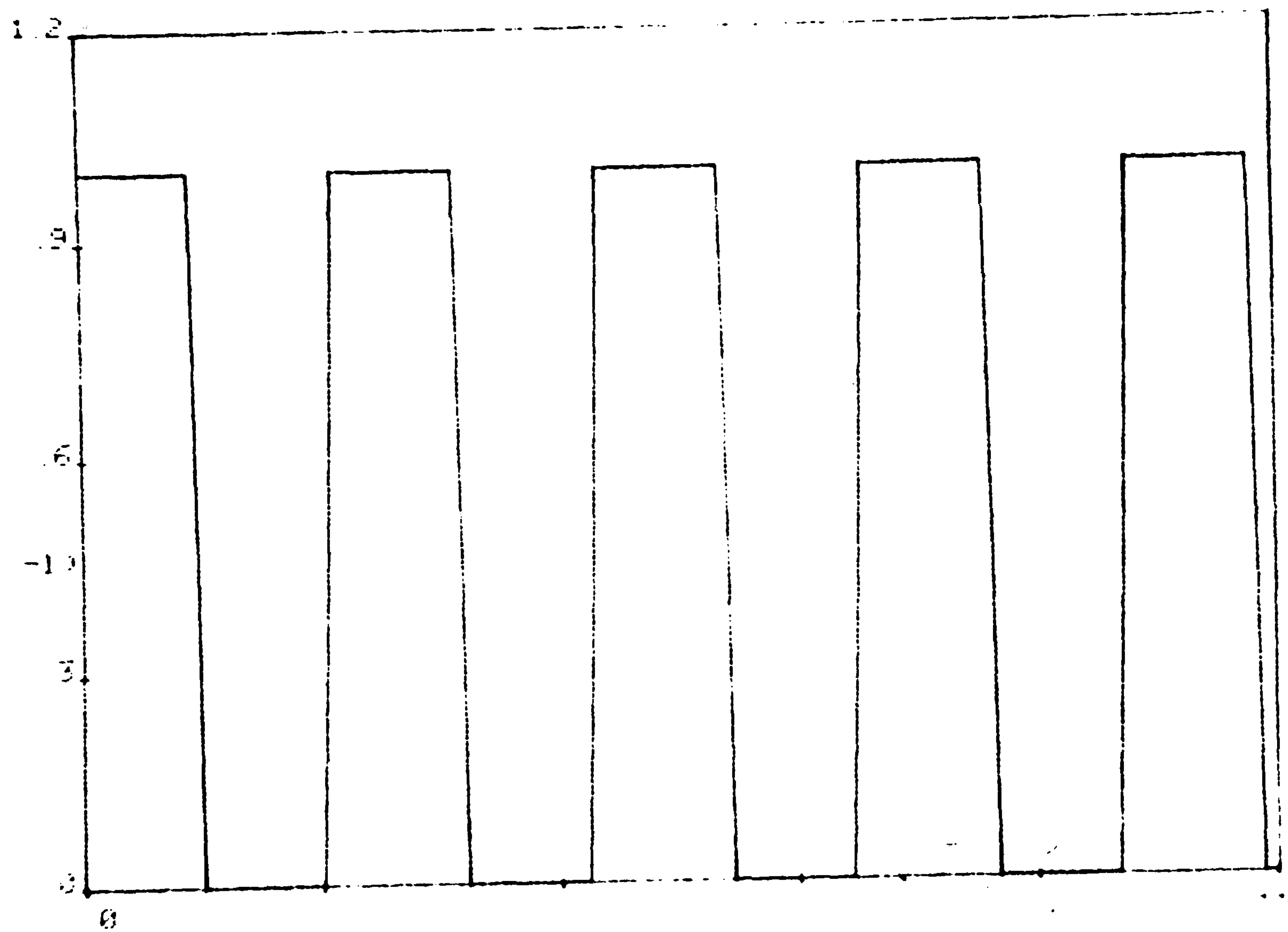


Figure 19. Plot for Input Function

quantized according to user requirements by either truncation or rounding. Corresponding coefficient values are shown in Table II. The designed coefficients appear in the first, the quantized coefficients in the second, and the scaled coefficients in the third columns of the table.

TABLE II

COEFFICIENT FOR DIRECT FORM

$\underline{b}$	$\hat{\underline{b}}_s$	$\underline{b}_s$
.1343790E 00	.9765625E-02	.1119825E-01
.2789370E 00	.2148438E-01	.2324475E-01
.3400000E 00	.2734375E-01	.2833333E-01
.2789370E 00	.2148438E-01	.2324475E-01
.1343790E 00	.9765625E-02	.1119825E-01

The expected output denoted by $\hat{y}_{exp}(n)$ can be calculated by using the equation below.

$$\hat{y}_{exp}(n) = \sum_{k=0}^{M} \hat{b}_{s_k} \hat{x}_s(n-k) \tag{4-1}$$

where $\hat{b}_s$ and $\hat{x}_s$ are the quantized and scaled coefficients; and M is the number of coefficients. The expected output for steady-state case is shown in Table XIII.

The actual output denoted by $\hat{y}_{act}(n)$ can be calculated by using the equation below. The above equation is very simular to Equation (4-1); however,

$$\hat{y}_{act}(n) = \sum_{k=0}^{M} \hat{b}_{s_k} \hat{x}_s(n-k) \tag{4-2}$$

The numbers used in Equation (4-2) are all binary. These numbers are shown in Table III. The first column is $\hat{x}_s$, the second, $\hat{b}_s$, and the third, $\hat{y}_{act}$.

TABLE III

BINARY NUMBERS RELATED TO EQUATION (4-2)

$\hat{\underline{x}}_s$	$\hat{\underline{b}}_s$	$\hat{\underline{y}}_{act}$
0000110011		000000000001111111100
0000110011		000000000110011000000
0000110011		000000001110000101000
0000110011		000000010010011101100
0000110011		000000010100011101000
0000110011		000000010100011101000
0000110011	0000000101	000000010100011101000
0000110011	0000001011	000000010100011101000
0000110011	0000001110	000000010100011101000
0000110011	0000001011	000000010100011101000
0000000000	0000000101	000000010010011101100
0000000000		000000001110000101000
0000000000		000000000110011000000
0000000000		000000000001111111100
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000

Corresponding real numbers in the first column in Table III will be used to plot the actual output which is shown in Figure 20. The actual output for steady-state is shown in Table XIII.

Cascade Form. As we mentioned in the previous chapter, the cascade form can be obtained by factoring the direct form transfer function. The digital filter studied

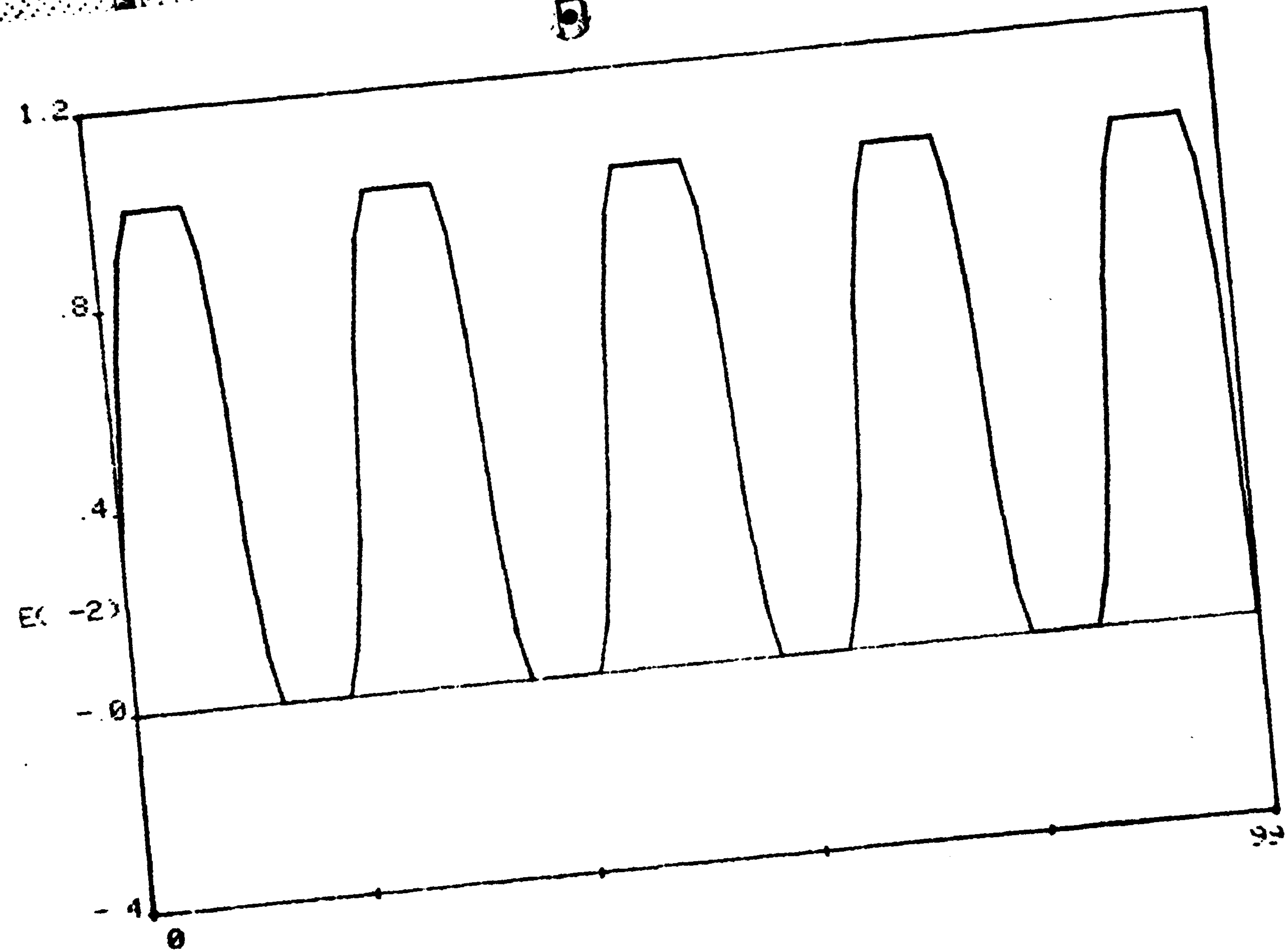


Figure 20. Direct Form Actual Output Response

for direct form can be factored into two second order digital filters. Corresponding coefficient values are shown in Table IV in the same way as in Table II for each second-order section.

TABLE IV

COEFFICIENTS FOR CASCADE FORM

a. First Second-Order Section

$\underline{b}$	$\hat{\underline{b}}_s$	b_s
.1000000E 01	.2539063E-01	.2631579E-01
.1777000E 01	.4492188E-01	.4676317E-01
.9999990E 00	.2539063E-01	.2631576E-01

b. Second Second-Order Section

b	$\hat{b}_s$	b_s
.1343790E 00	.3320313E-01	.3359476E-01
.4007000E-01	.9765625E-02	.1001750E-01
.1343790E 00	.3320313E-01	.3359476E-01

The steady-state expected and actual output for each second-order section can be calculated by using Equation (4-1) and (4-2), respectively. The number of coefficients denoted by M in both equations is two. The steady-state expected and actual output of the first second-order section will be the quantized input to the next second-order section. The steady-state expected and actual output of the last section will be the steady-state expected and actual output, respectively. The steady-state expected output is shown in Table XII and the corresponding binary

values of each second-order section input, coefficients, and actual output in Table V in the same way as in Table III.

Corresponding real numbers in the third column in Table Vb will be used to plot the actual output which is shown in Figure 21. The actual output for steady-state is shown in Table XIII.

Parallel Form. Each second-order section coefficients shown in Table IV are the same as for cascade form. The steady-state expected and actual output is also calculated in the same way. But the steady-state expected and actual output for parallel form will be the summation of the steady-state expected and actual output for each second-order section, respectively. The steady-state expected output is shown in Table XII and the corresponding binary number values for the second second-order section input, coefficients and actual outputs are shown in Table VI, using the same scheme as the one for Table III. The actual output of parallel filter is also shown in Table VI. The first second-order section binary number values are the same as shown in Table Vb.

Corresponding real numbers in Table VIb will be used to plot the actual output which is shown in Figure 22. The actual output for steady-state is shown in Table XIII.

Nested Form. The filter coefficients studied for direct form will be used to get the nested filter coefficient denoted by e_i using the following equation.

TABLE V

BINARY NUMBERS RELATED TO EQUATION (4-2) FOR CASCADE FORM

a. First Second-Order Section

$\hat{\underline{x}}$	$\hat{\underline{b}}_s$	$\hat{y}_{act}$
0000110011		000000000110110001100
0000110011		000000001000110001000
0000110011		000000001111100010100
0000110011		000000001111100010100
0000110011		000000001111100010100
0000110011		000000001111100010100
0000110011		000000001111100010100
0000110011		000000001111100010100
0000110011		000000001111100010100
0000110011	0000001101	000000001111100010100
0000110011	0000010111	000000001000110001000
0000000000	0000001101	000000000110110001100
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000

TABLE V (continued)

b. Second Second-Order Section

$\hat{x}_s$	$\hat{b}_s$	$\hat{y}_{act}$
0000000001		000000000000001101000
0000000010		000000000000100010100
0000000011		000000000000111111100
0000000011		000000000001010011110
0000000011		000000000001011111000
0000000011		000000000001011111000
0000000011		000000000001011111000
0000000011		000000000001011111000
0000000011	0000010001	000000000001011111000
0000000011	0000000101	000000000001011111000
0000000011	0000010001	000000000001011111000
0000000010		000000000001010011110
0000000001		000000000000111111100
0000000000		000000000000100010100
0000000000		000000000000001101000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000

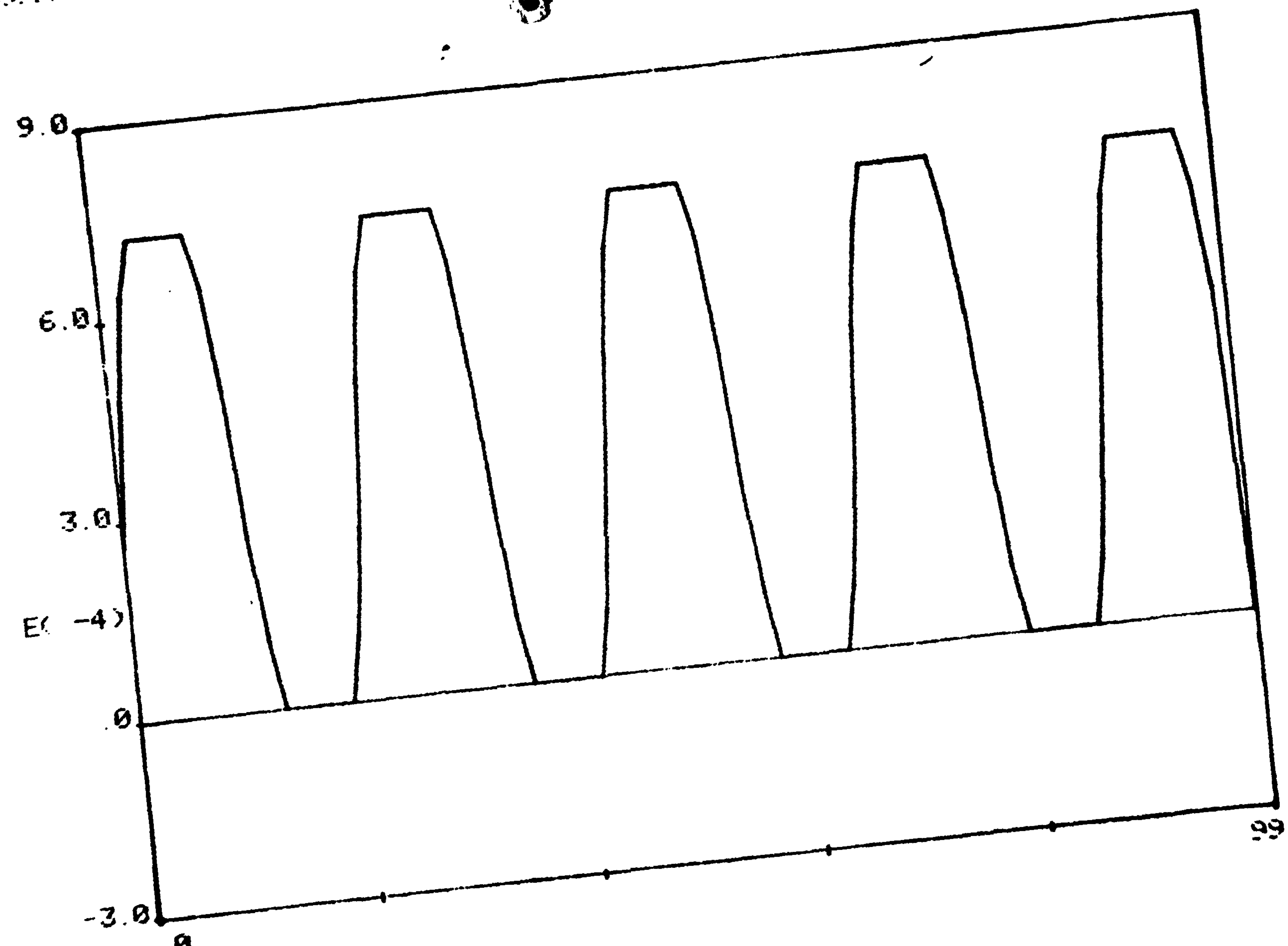


Figure 21. Cascade Form Actual Output Response

TABLE VI

BINARY NUMBERS RELATED TO EQUATION (4-2)
FOR PARALLEL FORM

$\hat{x}_s$	$\hat{b}_s$	$\hat{y}_{act}$
0000110011		000000000101001011100
0000110011		000000001111101010000
0000110011		000000010100110101100
0000110011		000000010100110101100
0000110011		000000010100110101100
0000110011		000000010100110101100
0000110011		000000010100110101100
0000110011	0000010001	000000010100110101100
0000110011	0000000101	000000010100110101100
0000110011	0000010001	000000010100110101100
0000000000		000000001111101010000
0000000000		000000000101001011100
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000

TABLE VI (continued)

b. Actual Output For Parallel Form

$\hat{y}_{act}$

```
000000000000001110011
000000000000011111011
000000000001000001000
000000000001010000001
000000000001100000010
000000000001100000010
000000000001100000010
000000000001100000010
000000000001100000010
000000000001100000010
000000000001100000010
000000000001010000001
000000000001000001000
000000000000011111011
000000000000001110011
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
```

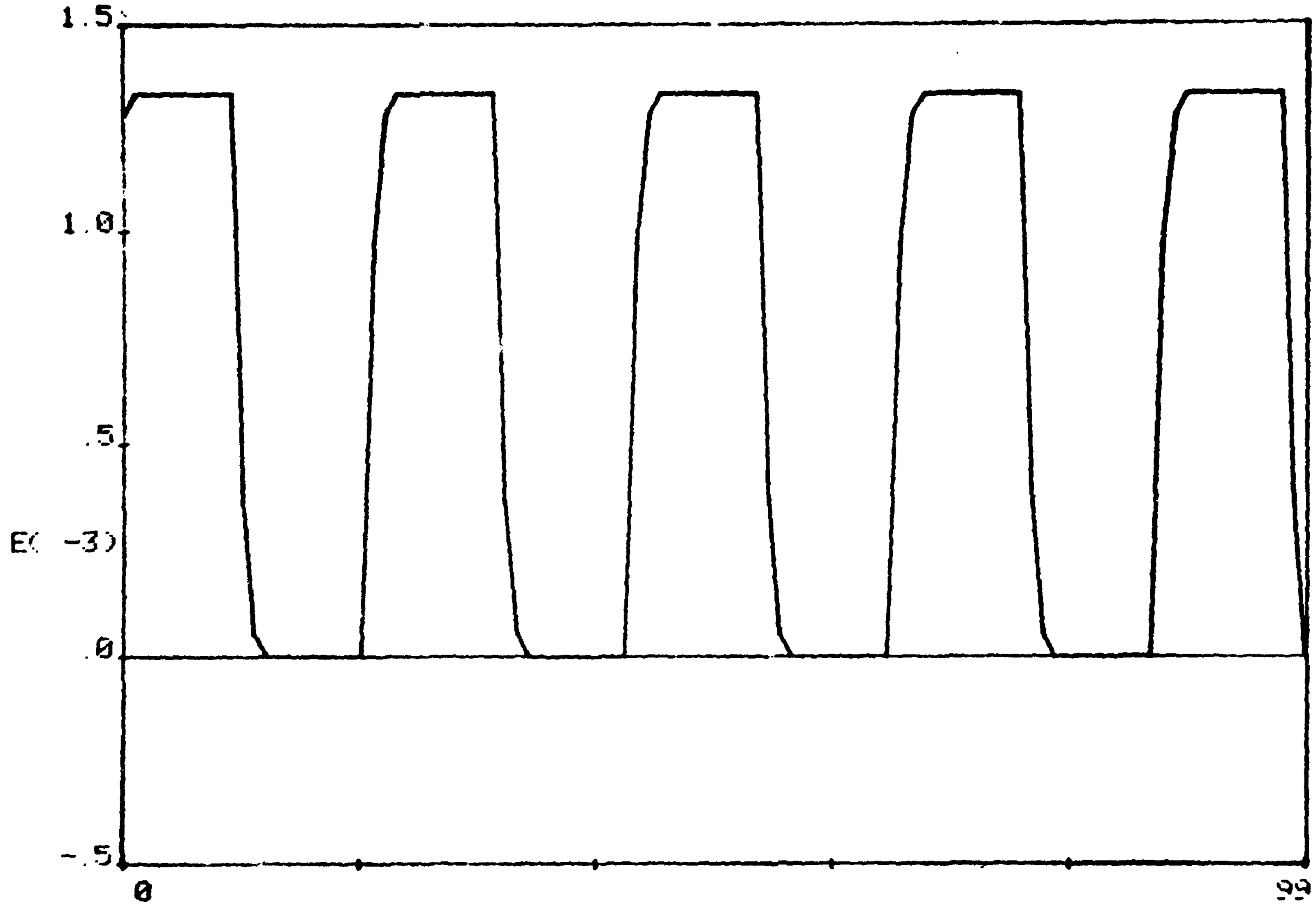
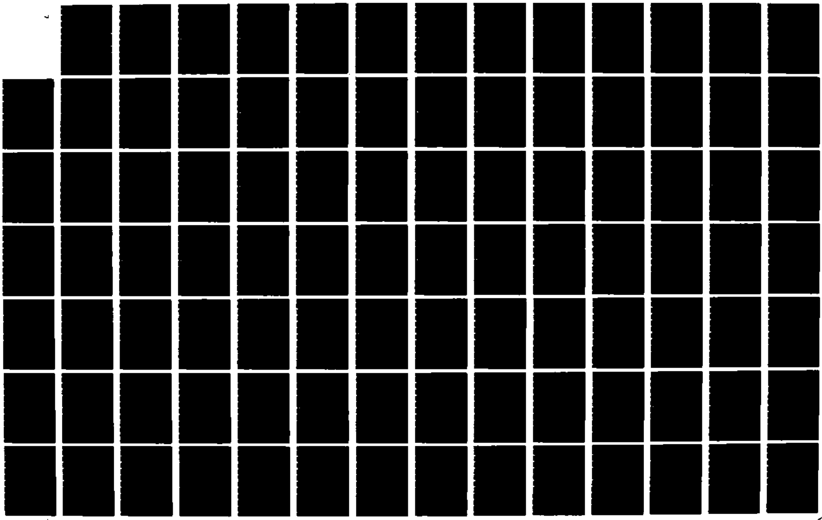
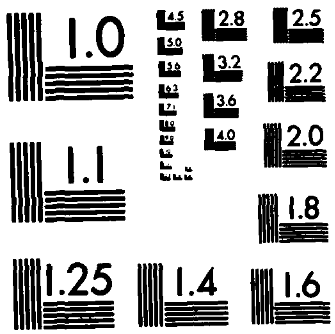


Figure 22. Parallel Form Actual Output Response

AD-A138 082 STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. H INANLI 2/3

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-32 F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

$$e_0 = b_{s_0}$$

$$e_n = \frac{b_{s_n}}{\hat{b}_{s_{n-1}}} \tag{4-3}$$

where b_s is the scaled coefficient in the direct form. Then, these coefficients will be scaled such that each coefficient is less than one-half the absolute maximum value of the coefficients in Equation (4-3) to prevent overflow. The nested filter scaled coefficients denoted by e_s are shown in Table VII.

TABLE VII

NESTED FILTER COEFFICIENTS

e_s
1.953125E-03
.500000
.275391
.177734
.109375

The expected and actual output can be calculated by using Equations (4-4) and (4-5) below, respectively.

$$\hat{y}_{exp}(n) = e_{s_0}(\hat{x}_s(n) + e_{s_1}(\hat{x}_x(n-1) + \ldots + e_{s_M}\hat{x}_s(n-M))\ldots) \tag{4-4}$$

$$\hat{y}_{act}(n) = e_{s_0}(\hat{x}_s(n) + e_{s_1}(\hat{x}_s(n-1) + \ldots + e_{s_M}\hat{x}_s(n-M))\ldots) \tag{4-5}$$

The expected output for steady-state is shown in Table XII. Corresponding binary number values for filter input, coefficients and actual output are shown in Table VIII in the same manner as in Table XIII.

TABLE VIII

BINARY NUMBERS RELATED TO EQUATION (4-5)
FOR NESTED FORM

$\hat{\underline{x}}_s$	$\underline{e}_s$	$\hat{\underline{y}}_{act}$
0000110011		000000000000011001100
0000110011		000000000000011111111
0000110011		000000000000100001000
0000110011		000000000000100001000
0000110011		000000000000100001000
0000110011		000000000000100001000
0000110011		000000000000100001000
0000110011	0000000001	000000000000100001000
0000110011	0100000000	000000000000100001000
0000110011	0010001101	000000000000100001000
0000000000	0001011011	000000000000000111100
0000000000	0000111000	000000000000000001001
0000000000		000000000000000000001
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000

Then, corresponding real numbers in the third column in Table VIII will be used to plot the actual output which is shown in Figure 23. The actual output for the steady-state case is shown in Table XIII.

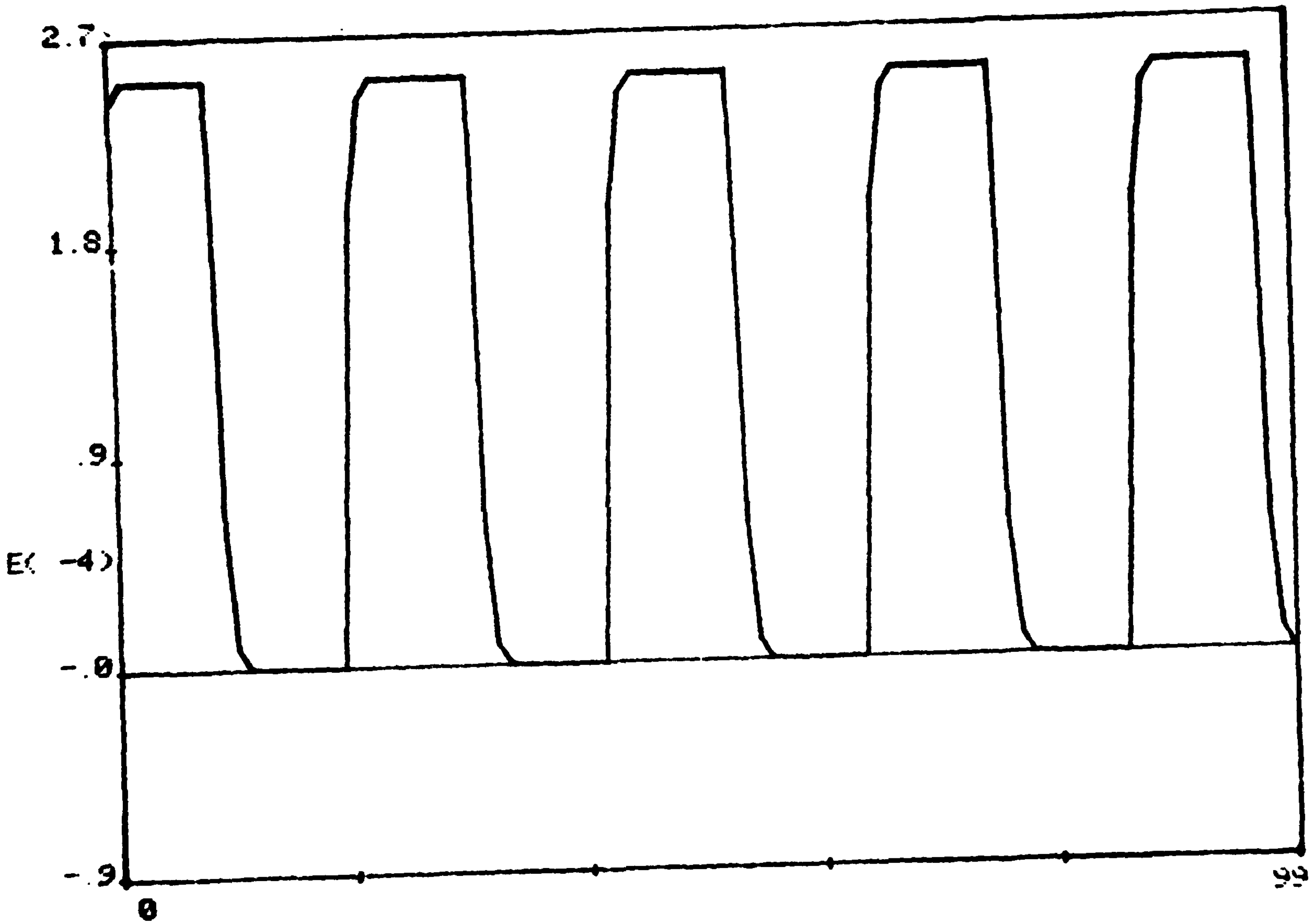


Figure 23. Nested Form Actual Output Response

Cascade-Nested Form. The coefficients studied for cascade form will be used to calculate the cascade-nested form coefficient in the same manner as in the nested form discussed above. The coefficients for each second-order section are shown in Table IX.

TABLE IX

COEFFICIENTS FOR CASCADE NESTED FORM

a. First Second-Order Section

$\underline{e}_s$
3.906250E-03
4.296875E-02
.500000

b. Second Second-Order Section

$\underline{e}_s$
5.859375E-03
.500000
.158203

The steady-state expected and actual outputs can be calculated by letting M=2 in Equations (4-4) and (4-5), respectively. The expected output for the steady-state case is shown in Table XII. Corresponding binary number values for each second-order section input, coefficients and actual output are shown in Table X in the same manner as in Table III. Then the corresponding real numbers in the third column in Table Xb are used to plot the actual output which is shown in Figure 24. As we can see easily

TABLE X

BINARY NUMBERS RELATED TO EQUATION (4-5)
FOR CASCADE-NESTED FORM

a. First Second-Order Section

$\hat{\underline{x}}_s$	$\underline{e}_s$	$\hat{\underline{y}}_{act}$
0000110011		000000000000110011000
0000110011		000000000000110101001
0000110011		000000000000110110010
0000110011		000000000000110110010
0000110011		000000000000110110010
0000110011		000000000000110110010
0000110011		000000000000110110010
0000110011	0000000010	000000000000110110010
0000110011	0000010110	000000000000110110010
0000110011	0100000000	000000000000110011000
0000000000		000000000000110110010
0000000000		000000000000000011010
0000000000		000000000000000011010
0000000000		000000000000000011010
0000000000		000000000000000001000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
		000000000000000000000

TABLE X (continued)

b. Second Second-Order Section

$\hat{\underline{x}}_s$	$\underline{e}_s$	$\hat{\underline{y}}_{act}$
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000	0000000011	000000000000000000000
0000000000	0100000000	000000000000000000000
0000000000	0001010001	000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000

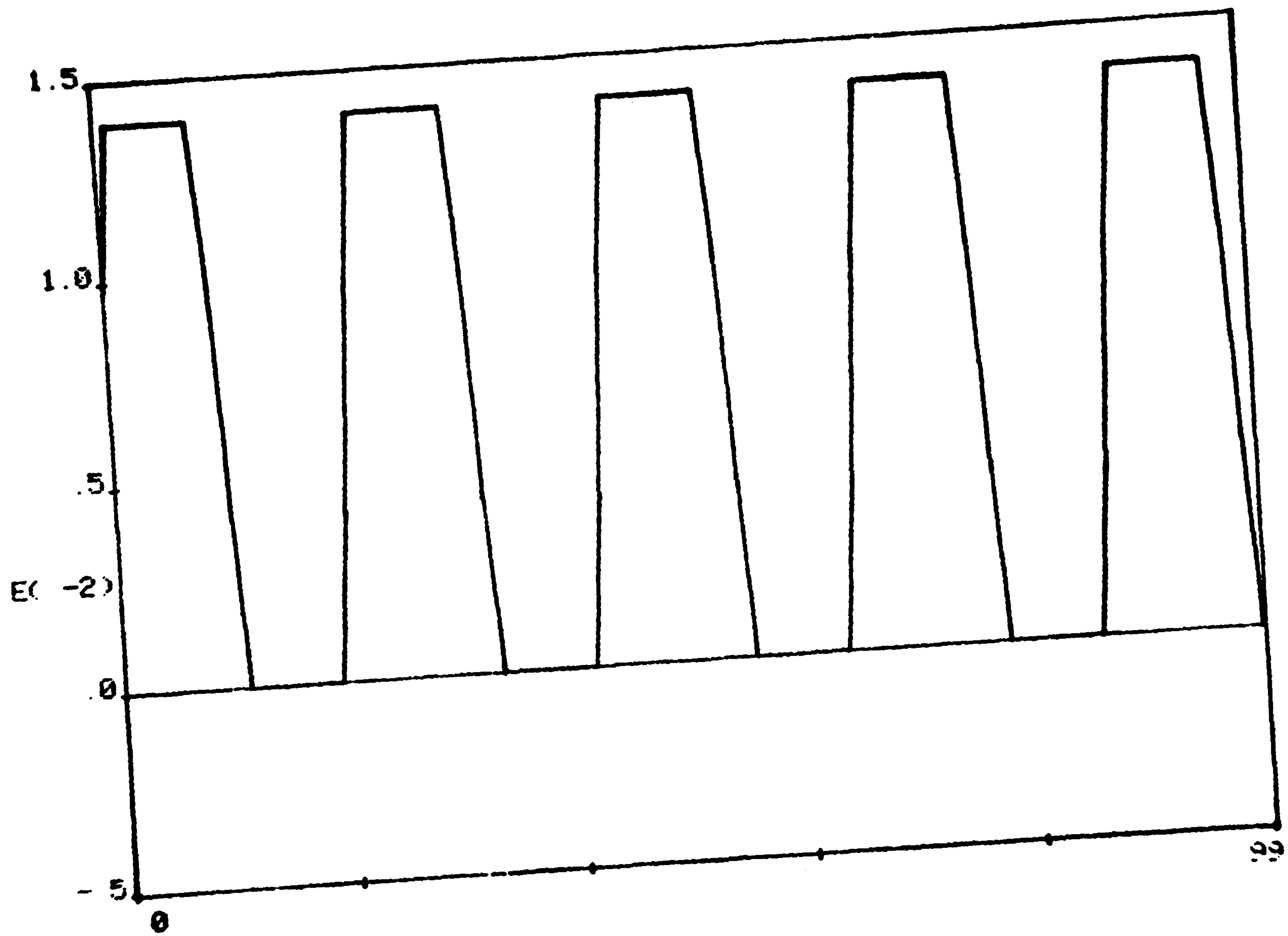


Figure 24. Parallel-Nested Form Actual Output Response

from Table X, the output of first second-order section is too small. Therefore, when it is quantized in accordance with the input word length, it will be all zero. So, the cascade-nested form will not give the actual output for short word length.

Parallel-Nested Form. Each second-order section coefficients shown in Table IX are the same as for cascade-nested form. The steady-state expected and actual outputs for parallel-nested form will be the summation of the steady-state expected and actual output for each second-order section, respectively. The steady-state expected output is shown in Table XII and the corresponding binary number values for the second second-order section input, coefficients and actual outputs are shown in Table XI. The actual output of parallel filter is also shown in Table XI. The first second-order section binary number values are the same as shown in Table Xa. Corresponding real numbers in Table XIb will be used to plot the actual output which is shown in Figure 24. The actual output for steady state is shown in Table XIII.

Finally, steady-state expected and actual outputs for all digital filters studied in this section are shown in Table XII and Table XIII, respectively.

TABLE XI

BINARY NUMBERS RELATED TO EQUATION (4-5)
FOR PARALLEL-NESTED FORM

a. Second Second-Order Section

$\underline{\hat{x}}_s$	$\underline{e}_s$	$\underline{\hat{y}}_{act}$
0000110011		000000000001001100100
0000110011		000000000001110010110
0000110011		000000000001111000110
0000110011		000000000001111000110
0000110011		000000000001111000110
0000110011		000000000001111000110
0000110011		000000000001111000110
0000110011		000000000001111000110
0000110011	0000000011	000000000001111000110
0000110011	0100000000	000000000001001100100
0000000000	0001010001	000000000001111000110
0000000000		000000000000101100010
0000000000		000000000000000110000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
0000000000		000000000000000000000
		000000000000000000000

TABLE XI (continued)

b. Actual Output for Parallel-Nested Form

$\hat{\underline{y}}_{act}$

```
000000000001111111100
000000000010100111110
000000000010101111000
000000000010101111000
000000000010101111000
000000000010101111000
000000000010101111000
000000000010101111000
000000000010101111000
000000000001111111100
000000000010101111000
000000000000101111100
000000000000000111000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
000000000000000000000
```

TABLE XII

STEADY-STATE $\hat{y}_{exp}(n)$ (10 bits)

Direct Form	.00894924
Cascade Form	.000726138
Parallel Form	.0171203
Nested Form	.00032389
Cascade-Nested Form	.00000383209
Parallel Nested Form	.000133572

TABLE XIII

STEADY-STATE $\hat{y}_{act}(n)$ (10 bits)

Direct Form	.008865625
Cascade Form	.0007247925
Parallel Form	.01396751
Nested Form	.00025177
Cascade-Nested Form	.00000000
Parallel-Nested Form	.001335144

Simulation II

The steady-state expected and actual output for all FIR filters are calculated in the same manner as in Simulation I, based on 16 bits word length. Since the longer word length is used, the quantized coefficients and the input values will be very close to the ideal values, assumed to be the scaled coefficients and the input. Table XIV and Table XV, arranged based on 16 bits word length, show the comparison with Table I and Table II, arranged based on 10 bits word length, respectively. Since the simulation procedure is identical to the one carried out for Simulation I, only the result will be shown in Tables XVI and XVII.

TABLE XIV

INPUT SEQUENCES BASED ON 16 BIT

$\underline{x}$	$\hat{\underline{x}}_s$	$\underline{x}_s$
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.1000000E 00	.9997559E-01	.1000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00
.0000000E 00	.0000000E 00	.0000000E 00

TABLE XV

COEFFICIENT FOR DIRECT FORM
BASED ON 16 BIT

b	$\hat{b}_s$	b_s
.1343790E 00	.1116943E-01	.1119825E-01
.2789370E 00	.2322388E-01	.2324475E-01
.3400000E 00	.2832031E-01	.2833333E-01
.2789370E 00	.2322388E-01	.2324475E-01
.1343790E 00	.1116943E-01	.1119825E-01

TABLE XVI

STEADY-STATE $\hat{y}_{exp}(n)$ (16 bits)

Direct Form	.00971404
Cascade Form	.000766406
Parallel Form	.017647
Nested Form	.000450728
Cascade-Nested Form	.00000384618
Parallel-Nested Form	.00134061

TABLE XVII

STEADY-STATE $\hat{y}_{act}(n)$ (16 bits)

Direct Form	.0096896
Cascade Form	.0007345751
Parallel Form	.01429798
Nested Form	.0003356934
Cascade-Nested Form	.000003637979
Parallel-Nested Form	.001395954

The ideal output represented by y_I can be calculated by using the Equation (4-6) for direct, cascade and parallel form and the Equation (4-7) for nested, cascade-nested and parallel-nested form shown below.

$$y_I(n) = \sum_{k=0}^{M} b_{s_k} x_s(n-k) \tag{4-6}$$

$$y_I(n) = e_0(x_s(n) + e_1(x_s(n-1) + \dots + e_M x_s(n-M))\dots) \tag{4-7}$$

where

x_s = scaled input

b_s = scaled coefficients

e = nested filter coefficients before it is quantized

Ideal-output responses for FIR filters studied here are shown in Table XVIII.

TABLE XVIII

STEADY-STATE $y_I(n)$

Direct Form	.00972191
Cascade Form	.000767394
Parallel Form	.0176601
Nested Form	.000391882
Cascade-Nested Form	.0000587236
Parallel-Nested Form	.00633241

If Table XVIII is compared with Tables XII, XIII, XVI, and XVII, it is obvious that as the word length is increased, the actual and expected output response is coming close to the ideal output response.

Deviation at the Output Response of the Digital Filter

Deviation is defined as the difference between the output responses of the digital filter based on the different word length. The expected and actual deviation of FIR filters studied here for 10 bits and 16 bits word length are shown in Tables XIX and XX.

TABLE XIX

EXPECTED DEVIATION

Direct Form	.0007114
Cascade Form	.000040297
Parallel Form	.000526715
Nested Form	.0001269
Cascade-Nested Form	.0000000461866
Parallel-Nested Form	.0000049

TABLE XX

ACTUAL DEVIATION

Direct Form	.000828
Cascade Form	.0000098
Parallel Form	.0003304
Nested Form	.0000953
Cascade-Nested Form	
Parallel-Nested Form	.0000608

Summary

The expected and actual outputs and deviation of the FIR digital filters studied in Chapter III are calculated and presented with tables based on 10 and 16 bits word length. The ideal output response is also presented.

V. Conclusion and Recommendations

In this thesis, we have considered the problem of finite word length effects in some special classes of digital filters. Some well-known and new structures have been presented for this case. For some of the new structures, the deviation at the output response remains constant or insignificant as the word length is increased.

One, who is interested in the low deviation at the output response due to finite word length registers, can find the result in Tables XIX and XX helpful. Corresponding output response of the digital filters is shown in Tables XII, XIII, XVI, XVII and XVIII. We can see from the tables that the digital filter, which has low deviation, gives very small output response which requires longer output register to recognize. As we know that it makes the arithmetic operation slower and increases the cost to use the longer register.

The techniques and software developed here can be used to evaluate other signal processing schemes in which binary operations with round-off and/or truncation are required, such as the FFT. The programs for fixed-point arithmetic in the Appendices can be extended for floating-point arithmetic. Thus, we may be able to determine the better arithmetic for a particular digital filter implementation. This work can be extended by studying other new

digital filter structures, and by studying in the same manner the IIR digital filters.

Appendix A

Flowgraph for Supporting the Desired Digital Filters

Appendix A contains the flowgraphs which help to understand the FORTRAN algorithm in Appendices B, C, and D. These flowgraphs are:

1. Decimal to Binary Number Converter
2. Two's Complement of Binary Numbers
3. Binary to Decimal Number Converter
4. Two's Complement Addition
5. Binary Multiplication
6. Shift-left and Shift-right Operator
7. FIR Direct Form Structure
8. FIR Cascade Form Structure
9. FIR Parallel Form Structure
10. FIR Nested Form Structure
11. FIR Cascade-Nested Form Structure
12. FIR Parallel-Nested Form Structure

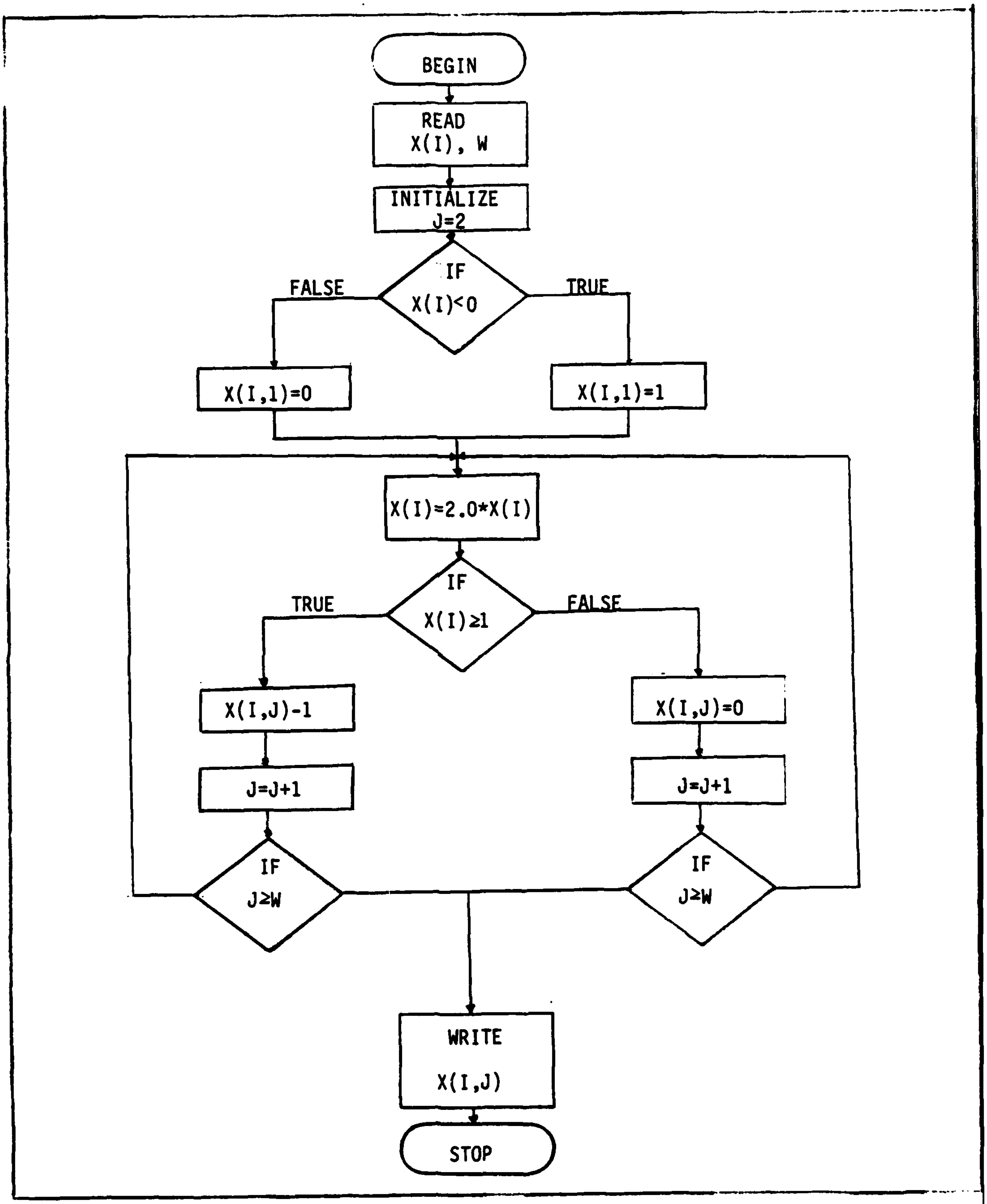


Figure 25. Decimal to Binary Numbers Converter

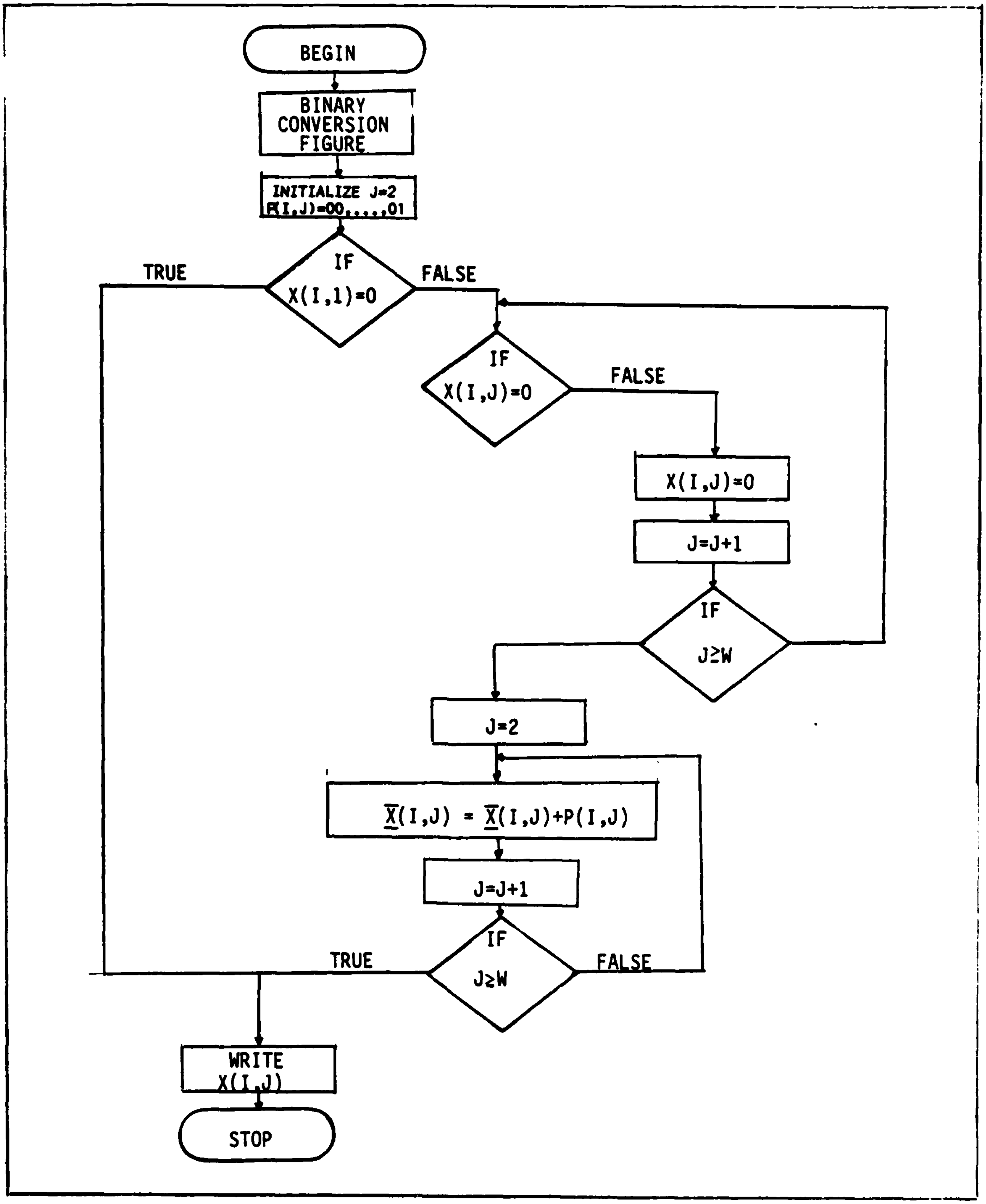


Figure 26. Two's Complement of Binary Numbers

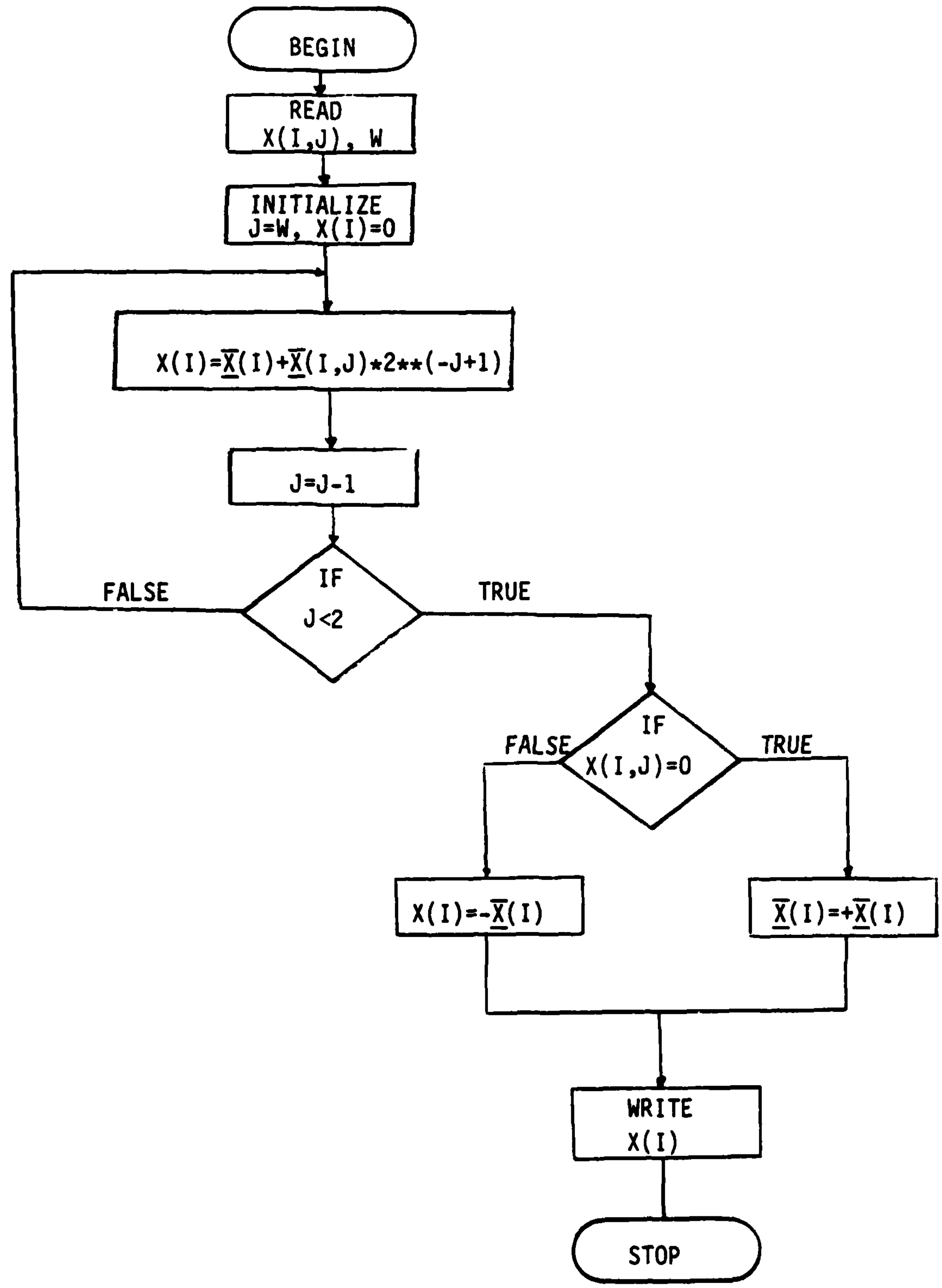


Figure 27. Binary to Decimal Number Converter

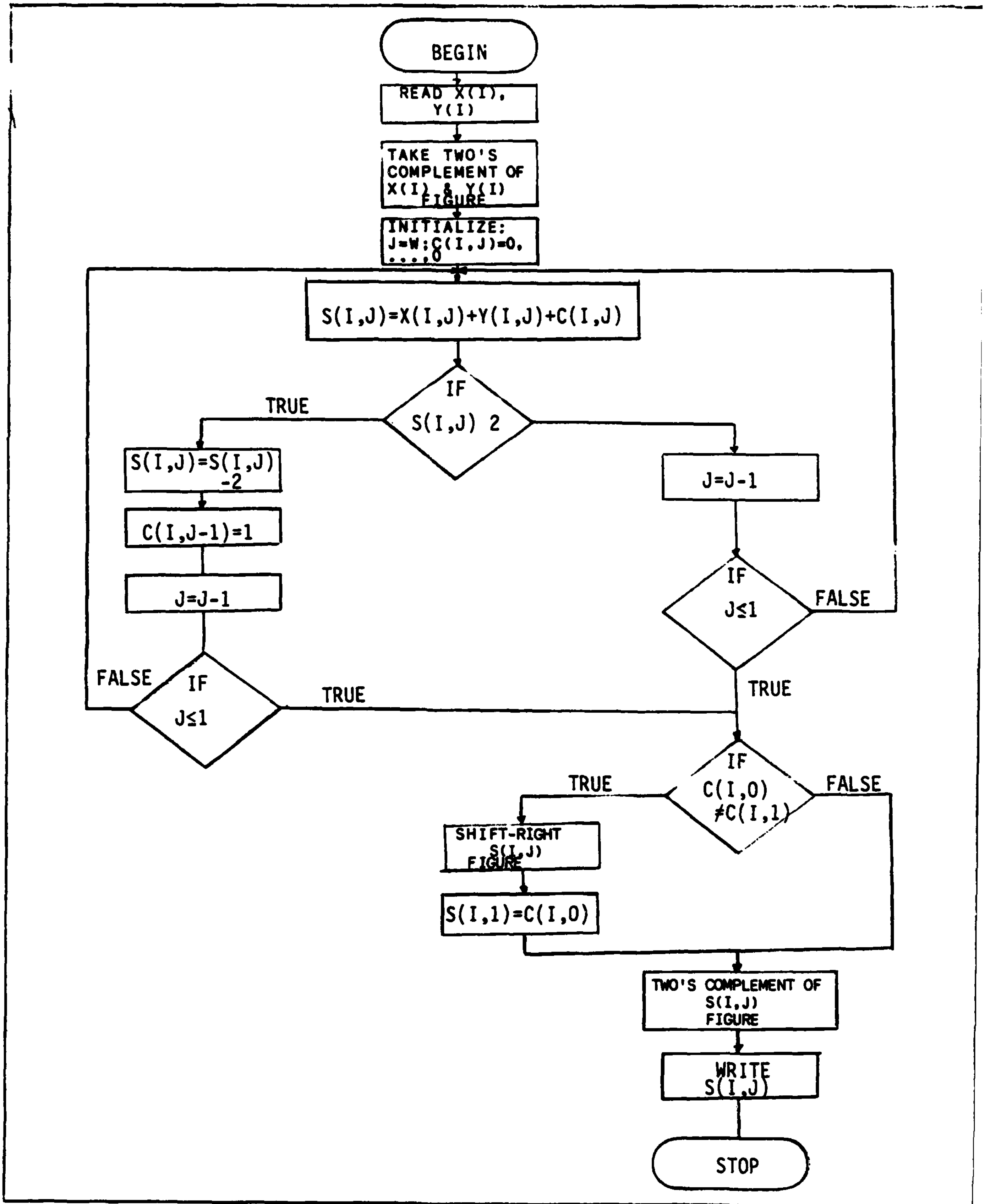


Figure 28. Two's Complement Addition

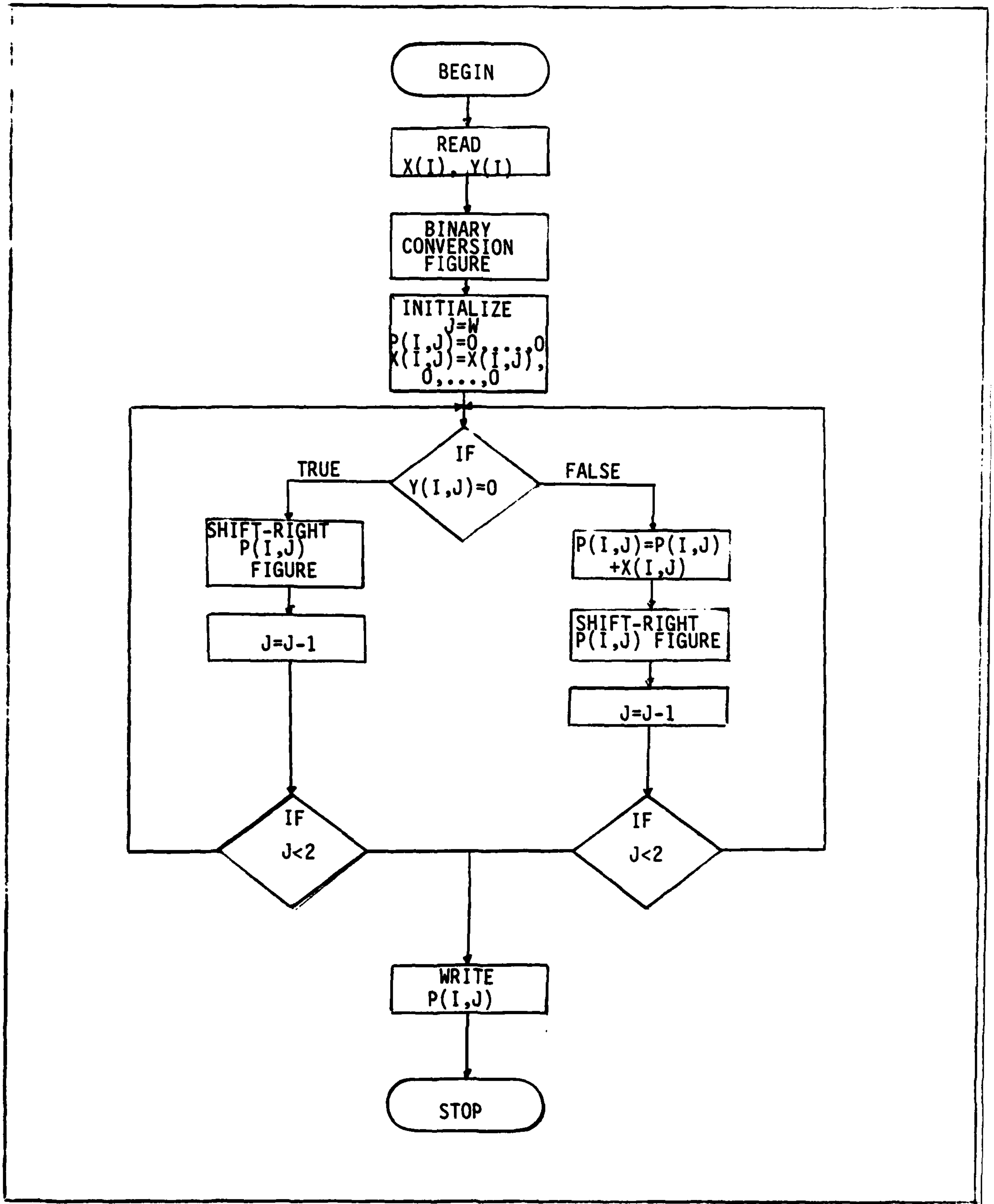


Figure 29. Binary Multiplication

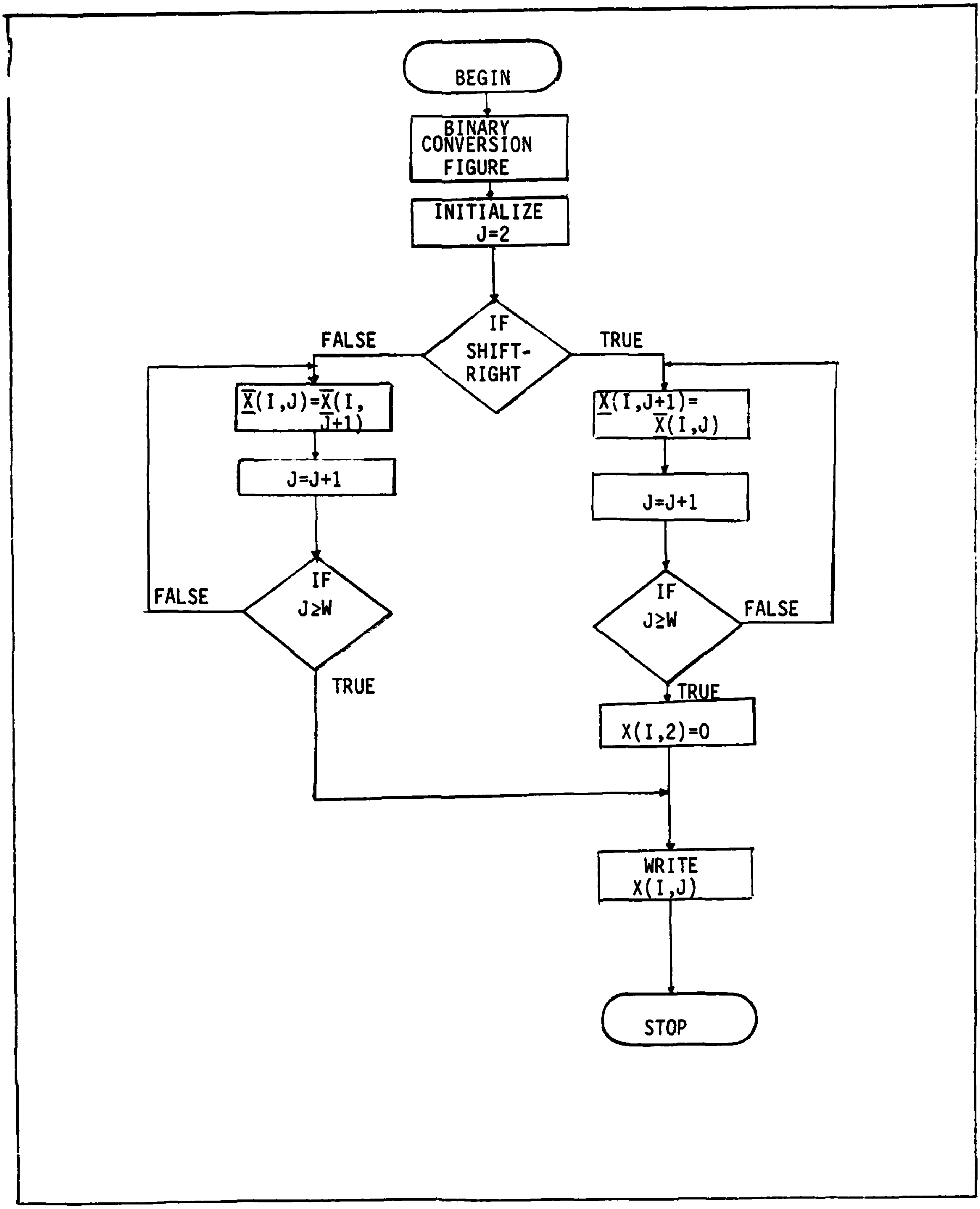


Figure 30. Shift-left and Shift-right Operator

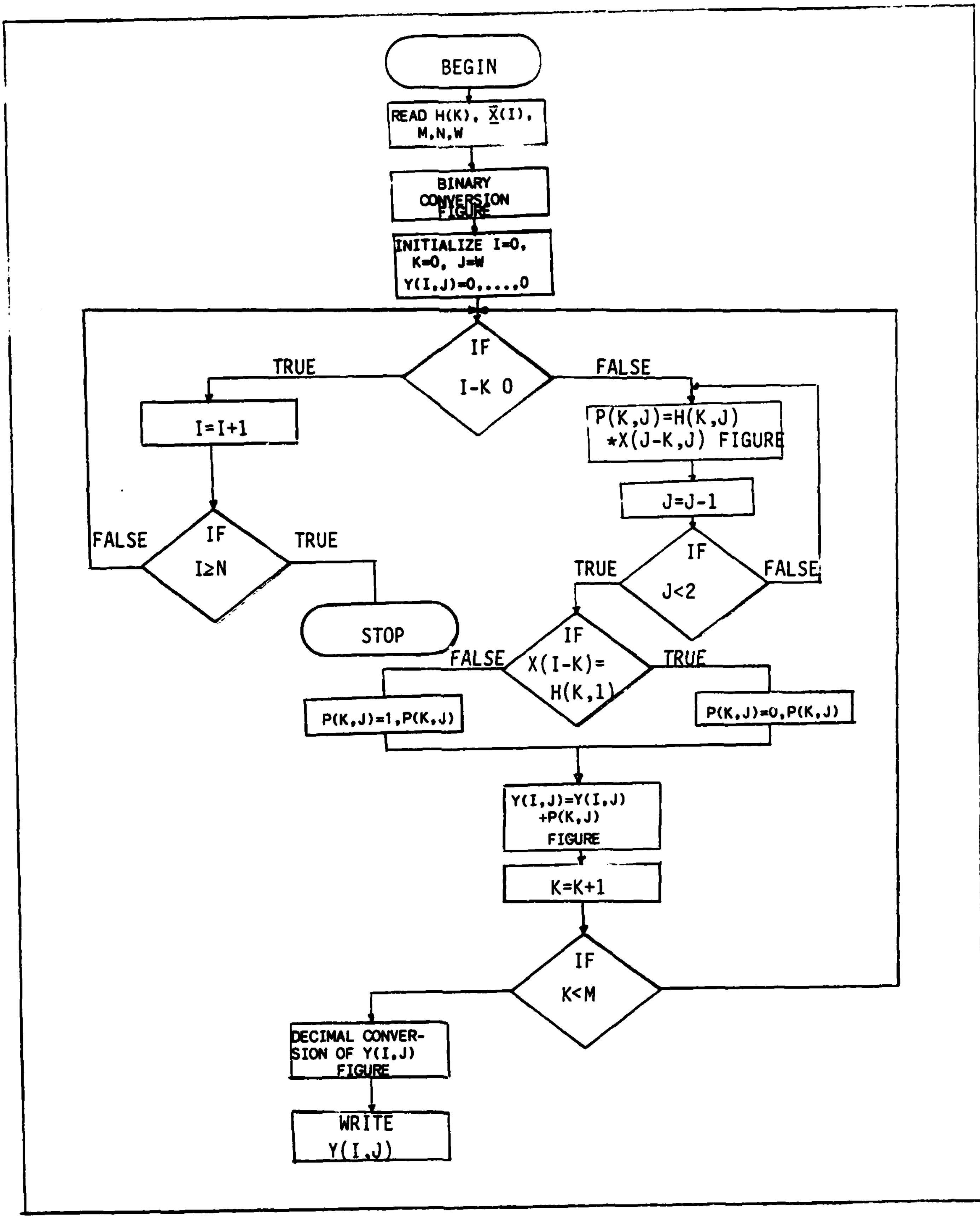


Figure 31. FIR Direct Form Structure

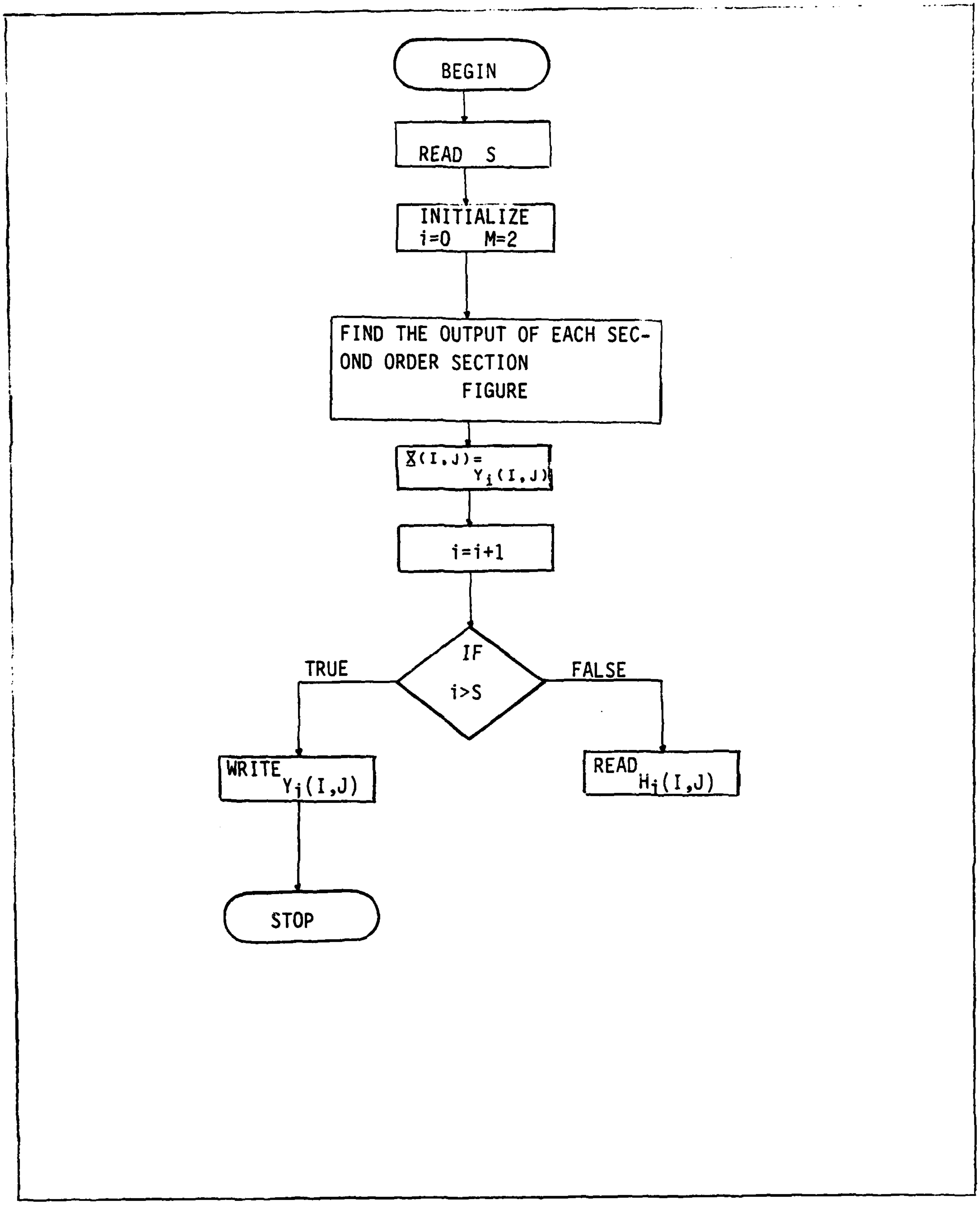


Figure 31. FIR Cascade Form Structure

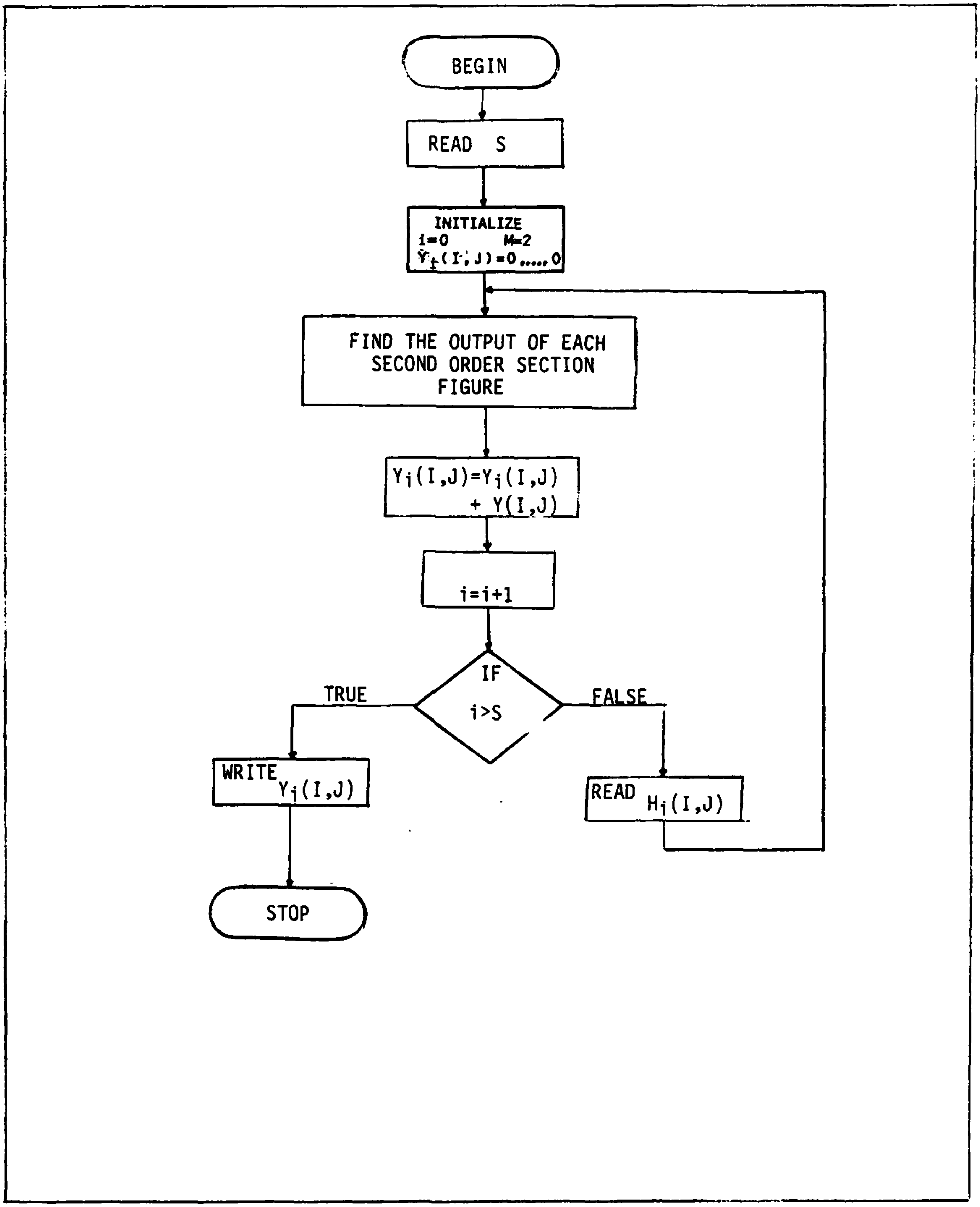


Figure 33. FIR Parallel Form Structure

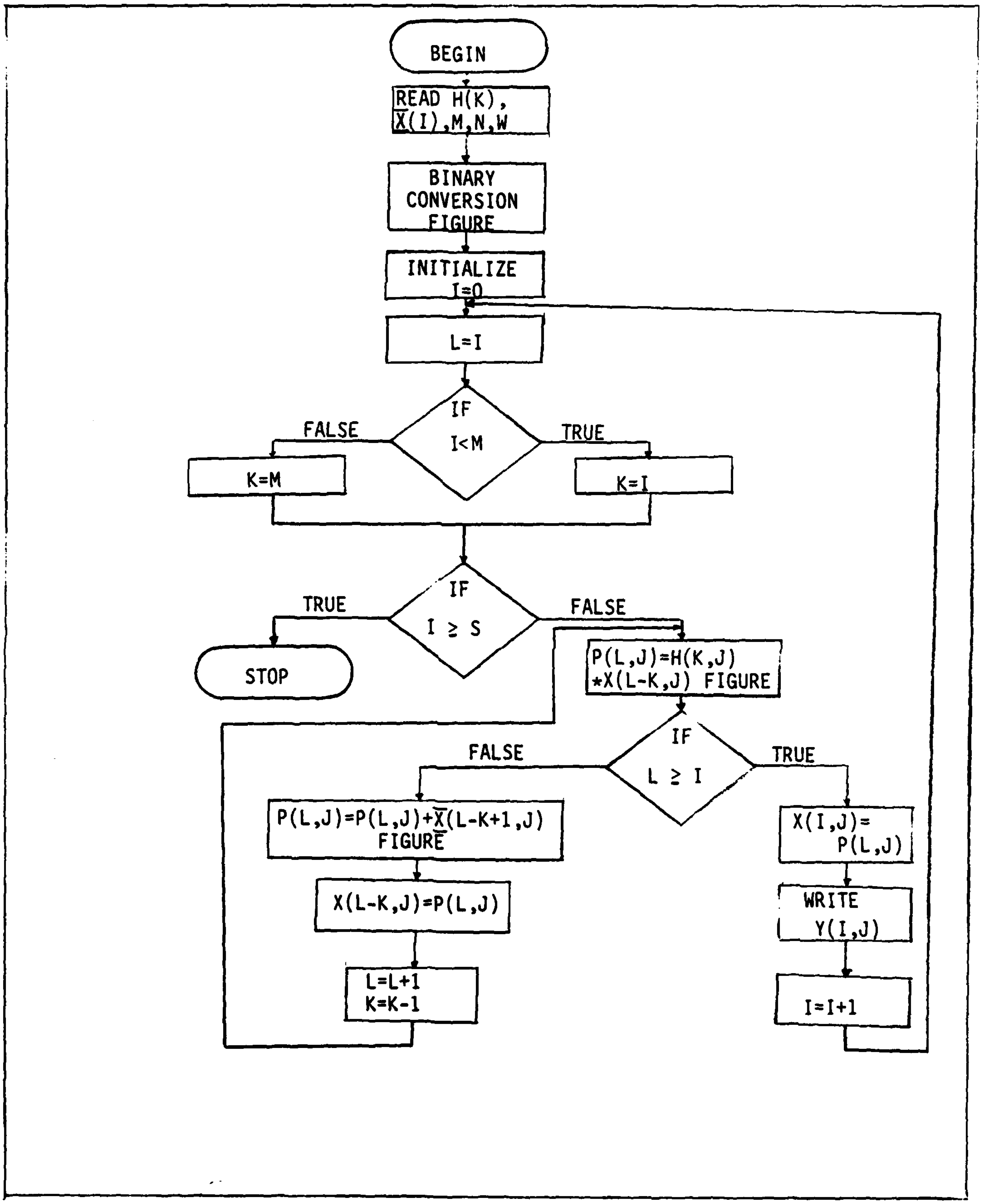


Figure 34. FIR Nested Form Structure

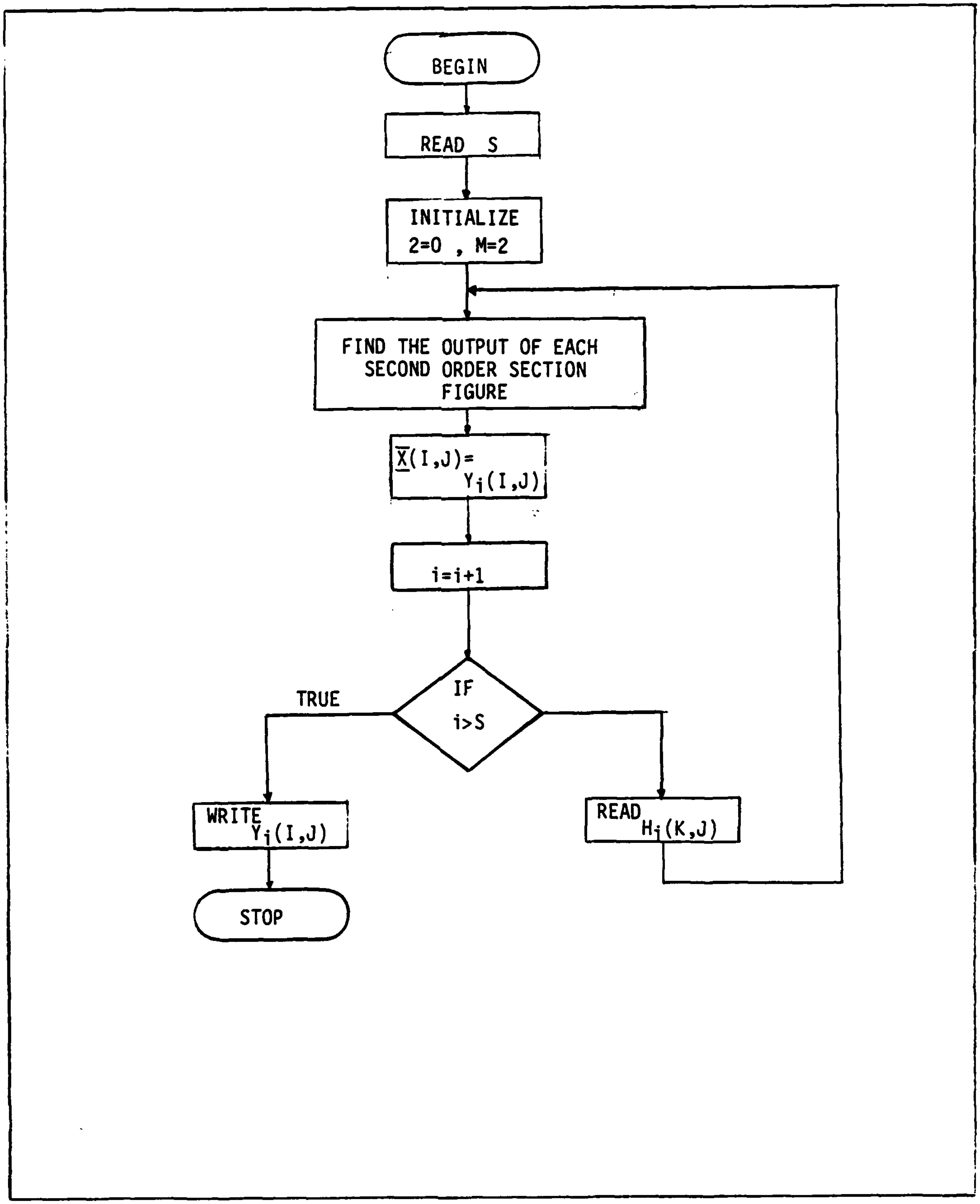


Figure 35. FIR Cascade-Nested Form Structure

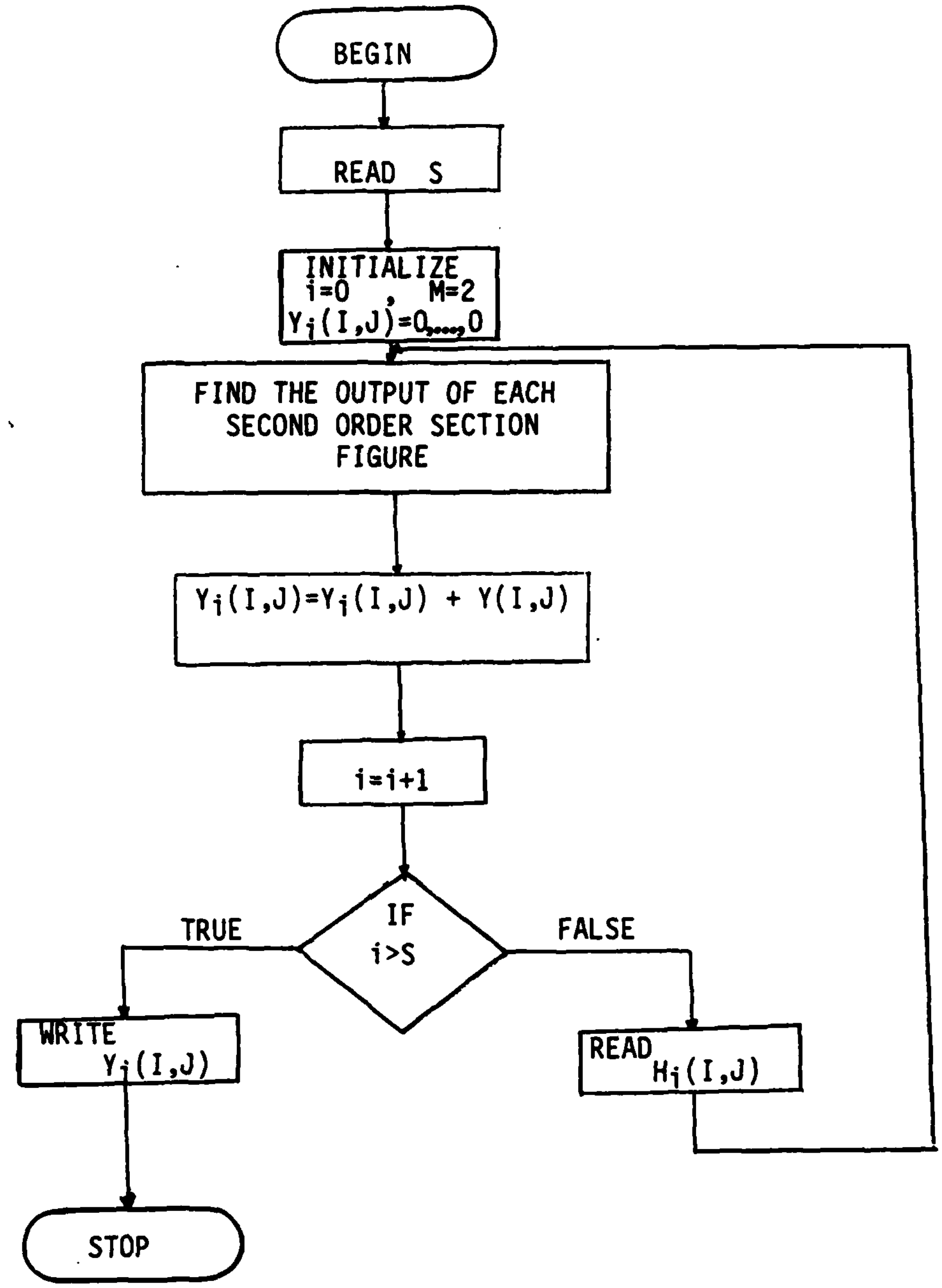


Figure 36. FIR Parallel-Nested Form Structure

Appendix B

Coefficients and Input to the Digital Filters

Appendix B contains the program, which can scale and quantize the coefficients and the input for the digital filter, and user's manual. Each program's user manual explains what the program does. These are called as follows:

1. IN.FR
2. NEWC
3. NES1
4. HA

USER'S MANUAL PROGRAM IN.FR

FILE:	IN.FR
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Scaling and quantization of given filter coefficients.
FUNCTION:	This program first reads the filter coefficients from the file. Then, it scales those coefficients such that the summation of the absolute value of the coefficients is less than 0.1. Finally, it quantizes these coefficients according to user requirements of either the truncation or the rounding technique.
PROGRAM USE:	The program is loaded by the following command:

RLDR IN IN1 IN3 IN4 @FLIB@

SUBROUTINE REQUIRED:

Name	Location	Purpose
IN1.FR	DP4:OWEN	To read the filter coefficient
IN3.FR	DP4:OWEN	To scale the filter coefficient
IN4.FR	DP4:OWEN	To quantize the filter coefficient

EXECUTION OF THE PROGRAM AND ITS OUTPUT FOLLOWS:

```
IN
FILTER COEFFICIENT FILE NAME:  FC
ENTER FILE NAME:  TC
COEFFICIENT FILE NAME FOR PLOT:  TC1
WORD LENGTH:  16
QUANTIZATION TYPE (1-TRUNCATION, 0-ROUNDING) 1
```

The input data file called FC contains the coefficients according to the equation shown below:

$$H(z) = A0 \frac{B(0) + B(1)z^{-1} + \ldots + B(M)z^{-M}}{A(0) + A(1)z^{-1} + \ldots + A(M)z^{-M}}$$

File FC is presenting the necessary data as shown below:

<u>FC</u>

```
5
0
3.934541E-02
 .210533
 .341118
 .341118
 .210533
3.934541E-02
1.00000
1.00000
```

where M=5, N=0, B(0)=3.934541E-02, ..., B(5)=3.934541E-02, A(0)=1.00000 and A0=1.00000.

File TC stores the coefficients (in binary) after they are scaled.

<u>TC</u>

```
16
6
0000000001101011
0000001000111110
0000001110100011
0000001110100011
0000001000111110
0000000001101011
```

where 16 desired number of bits in coefficient register, 6 is the number of coefficient.

File TC1 stores both quantized and scaled coefficients as well as the coefficients coming from file FC. The first column shows the coefficient numbers; the second, the coefficients coming from file FC; the third, quantized coefficients and the fourth, the scaled coefficients in file TC1.

TC1

5

1	.1343790E 00	.9765625E-02	.1119825E-01
2	.2789370E 00	.2148438E-01	.2324475E-01
3	.3400000E 00	.2734375E-01	.2833333E-01
4	.2789370E 00	.2148438E-01	.2324475E-01
5	.1343790E 00	.9765625E-02	.1119825E-01

USER'S MANUAL SUBROUTINE IN1.FR

FILE:	IN1.FR
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Reading of given filter coefficients.
FUNCTION:	This subroutine reads the given filter coefficients from the file.
SUBROUTINE REQUIRED:	None

USER'S MANUAL SUBROUTINE IN3.FR

FILE:	IN3.FR
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Scaling of given filter coefficients.
FUNCTION:	This subroutine scales the given filter coefficients such that the summation of the absolute value of the coefficients is less than 0.1.
SUBROUTINE REQUIRED:	None

USER'S MANUAL SUBROUTINE IN4.FR

FILE:	IN4.FR
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Quantization of digital filter coefficients.
FUNCTION:	This subroutine quantizes the scaled digital filter coefficients according to user requirements of either the truncation or the rounding technique. First, the scaled coefficient is converted into binary and placed in the coefficient register. The coefficient register can be a maximum of 140 bits long. Then, according to user

requirements, this binary number is truncated or rounded to the desired word length. Finally, the quantized number is converted back to the real number and stored in the file.

SUBROUTINE REQUIRED: None

FLOWGRAPH:

Type	Figure
1. Decimal to Binary Number Converter	25
2. Two's Complement of Binary Numbers	26
3. Binary to Decimal Converter	27

```
C***********************************************************************
C       PROGRAM :        IN.FR
C       AUTHOR   :       HARUN INANLI
C       DATE     :       SEPTEMBER 83
C       LANGUAGE:        FORTRAN 5
C
C       FUNCTION:        THIS PROGRAM IS USED TO SCALE AND QUANTIZE
C                        THE FILTER COEFFICIENT IN EITHER TRUNCATION
C                        OR ROUNDING TECHNIQUE ACCORDING TO USER REQUIRMENT
C                        THE FILTER COEFFICIENT IS OPTAINED BY USING THE
C                        PROGRAM CALLED WFILTER. QUANTIZED FILTER COEFFICIE
C                        IS STORED IN THE FILE NAMED BY THE USER IN BINERY
C
C***********************************************************************
        DIMENSION B(500),A(500)
        DIMENSION OUTFILE(7),H(500)
        DIMENSION FF(70),HH(70),MM(70),NN(70),SS(70),BA(500),DD(500)
        DIMENSION DD(500),B1(500)..
        INTEGER FF,HH,MM,NN,SS,W ,K
        CALL IN1(OUTFILE,B,A,M,N,AO)
        CALL IN3(B,M,B1)
        CALL IN4(B1,M,B)
        STOP
        END
```

```
C***********************************************************************
C
C       PROGRAM :        IN1.FR
C       AUTHOR   :       HARUN INANLI
C       DATE     :       SEPTEMBER 83
C       LANGUAGE:        FORTRAN 5
C
C       FUNCTION:        THIS PROGRAM IS USED TO READ THE FILTER
C                        COEFFICIENT PRODUCED BY DESIGN PROGRAM
C                        WFILTER ACCORDING TO USER REQUIREMENT.
C
C***********************************************************************
        SUBROUTINE IN1(OUTFILE,B,A,M,N,AO)
        DIMENSION OUTFILE(7),B(500),A(500)
        ACCEPT "FILTER COEFFICIENTS FILE NAME : "
        READ(11,10)OUTFILE(1)
   10   FORMAT(S15)
        CALL OPEN(1,OUTFILE,1,IER)
        IF (IER.NE.1)TYPE "OPEN ERROR",IER
        READ FREE(1)M
        READ FREE(1)N
        READ FREE(1) (B(I),I=1,M+1)
        READ FREE(1) (A(I),I=1,N+1)
        READ FREE(1)AO
        CALL CLOSE(1,IER)
        IF (IER.NE.1) TYPE "CLOSE FILE ERROR",IER
        RETURN
        END.
```

```
C*********************************************************************
C
C        PROGRAM :          IN3.FR
C        AUTHOR  :          HARUN INANLI
C        DATE    :          SEPTEMBER 83
C        LANGUAGE:          FORTRAN 5
C
C        FUNCTION:          THIS SUBROUTINE IS USED TO SCALE THE FILTER
C                           COEFFICIEN SUCH THAT THE SUMMATION OF THE
C                           ABSULUTE VALUE OF THE COEFFICIENTS IS LESS
C                           THAN (0.1).
C
C*********************************************************************
        SUBROUTINE IN3(B,M,B1)
        DIMENSION B(500),BA(500),B1(500)
        REAL SUM
        INTEGER K
        L=1000
        DO 20 K=1,L
          SUM=0
          DO 30 I=1,M+1
            BA(I)=ABS(B(I))
            BA(I)=BA(I)/K
            SUM=SUM+BA(I)
 30       CONTINUE
          IF(SUM.LT.(.1))GO TO 50
 20     CONTINUE
 50     CONTINUE
        DO 52 I=1,M+1
 52       B1(I)=B(I)/K
        RETURN
        END
```

```
C*****************************************************************
C
C         PROGRAM :          IN4.FR
C         AUTHOR  :          HARUN INANLI
C         DATE    :          SEPTEMBER 83
C         LANGUAGE:          FORTRAN 5
C
C         FUNCTION:          THIS SUBROUTINE IS USED TO QUANTIZE THE FILTER
C                            COEFFICIENTES IN EITHER TRUNCATION OR ROUNDING
C                            TECHNIQUE ACCORDING TO USER REQUIERMENT. THEN
C                            CALCULATE THE QUANTIZE ERROR AND STORE ALL
C                            THESE DATA IN THE FILE.
C
C*****************************************************************
      SUBROUTINE IN4(B1,M,B)
      DIMENSION B(500),BK(500),D(500),BN(500),BB(500)
      DIMENSION BC(500),BA(500),BD(500),DD(500),B1(500)
      INTEGER OUTFILE(7),OUTF(5)
      INTEGER HH(70),K,MM(70),NN(70),FF(70),OPT,SS(70)
      INTEGER W
      ACCEPT"ENTER FILE NAME : "
      READ(11,400)OUTFILE(1)
 400  FORMAT(S13)
      CALL DFILW(OUTFILE,IER)
      IF(IER.EQ.13) GO TO 401
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
 401  CALL CFILW(OUTFILE,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(1,OUTFILE,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      ACCEPT"COEFFICIENT FILE NAME FOR PLOT : "
      READ(11,900)OUTF(1)
 900  FORMAT(S15)
      CALL DFILW(OUTF,IER)
      IF(IER.EQ.13)GO TO 910
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
 910  CALL CFILW(OUTF,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      ACCEPT"WORD LENGTH : ",W
      ACCEPT"QUANTIZATION TYPE (1-TRUNCATION,0-ROUNDING)",OPT
      A=W-1
      AA=W+1
      A1=A-1
      DO 56 L=1,AA
        HH(L)=0
        FF(L)=0
        NN(L)=0
        SS(L)=0
        MM(L)=0
 56   CONTINUE
```

```
      IF(OPT.EQ.1)GO TO 11
      IF(OPT.EQ.0)GO TO 91
C*******************************************************
C
C     TRUNCATION OPTION
C
 11   DO 10 I=1,M+1
        IF(B1(I).LT.(0.0))GO TO 81
        HH(1)=0
        GO TO 82
 81     HH(1)=1
 82     BB(I)=2.0*ABS(B1(I))
C********* THE LOOP 20 IS USED TO CONVERT THE***************
C                   DECIMEL NUMBER TO BINERY.
        DO 20 K=2,W
          IF(BB(I).GE.1.0)GO TO 30
          HH(K)=0
          GO TO 40
 30       HH(K)=1
          BB(I)=BB(I)-1.0
 40       BB(I)=BB(I)*2.0
 20     CONTINUE
C********* END OF LOOP 20 ****************************
        BK(I)=0.0
C********* THE LOOP 60 IS USED TO CONVERT THE ***************
C                   BINERY NUMBER TO DECIMEL.
        DO 60 K=2,A
 60       BK(I)=BK(I)+HH(K)*(2.0**(-K+1))
C********** END OF LOOP 60 ******************************
        IF(HH(1).EQ.1)GO TO 100
        BN(I)=BK(I)
        GO TO 110
 100    BN(I)=-BK(I)
 110    D(I)=B1(I)-BN(I)
 10   CONTINUE
C******** THE INFORMATION OPTAINED ABOVE IS STORED IN FILE ******
      WRITE(10,200)W
      WRITE(1,500)W
      WRITE(1,500)(M+1)
      WRITE(2,500)(M+1)
      WRITE(10,201)
      WRITE(10,202)
      WRITE(10,203)
 500  FORMAT(20X,I5)
 200  FORMAT(4X,"WORD LENGTH : ",I4)
 201  FORMAT(4X,"USED QUANTIZATION TYPE IS TRUNCCTION")
 202  FORMAT(4X,"I",3X,"COEFFICIENT B(I)",9X,"SCALED COEFFICIENT"
     1          ,5X,"ROUNDOFF ERROR")
 203  FORMAT(4X,"-",3X,"----------------",9X,"------------------"
     1          ,5X,"--------------")
      DO 204 I=1,M+1
        WRITE(10,205)I,B(I),B1(I),D(I)
        WRITE(2,901)I,B(I),B1(I),D(I)
```

```
 204    CONTINUE
 205    FORMAT(1X,I4,2X,E14.7,14X,E14.7,6X,E14.7)
 901    FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E14.7)
        WRITE(10,206)
 206    FORMAT(1X,"TRUNCATED COEFFICIENT IN BINARY")
        DO 230 L=1,AA
 230      HH(L)=0
        DO 207 I=1,M+1
          IF(B1(I).LT.(0.0))GO TO 208
          HH(1)=0
          GO TO 209
 208      HH(1)=1
 209      BB(I)=2.0*ABS(B1(I))
          DO 210 K=2,W
            IF(BB(I).GE.1.0)GO TO 211
            HH(K)=0
            GO TO 212
 211        HH(K)=1
            BB(I)=BB(I)-1.0
 212        BB(I)=2.0*BB(I)
 210      CONTINUE
          WRITE(10,213)(HH(K),K=1,W)
          WRITE(1,213)(HH(K),K=1,W)
 213      FORMAT(12X,70(I1))
 207    CONTINUE
        GO TO 55
C
        END OF TRUNCATION OPTION
C
C***************************************************
C****************************************************
C
C       ROUNDING OPTION
C
 91     DO 26 I=1,(M+1)
          IF(B1(I).LT.(0.0))GO TO 21
          FF(1)=0
          GO TO 22
 21       FF(1)=1
 22       BC(I)=2.0*ABS(B1(I))
C********THE LOOP 23 IS USED TO CONVERT THE ****************
C                  DECIMEL NUMBER TO BINERY.
          DO 23 K=2,AA
            IF(BC(I).GE.1.0)GO TO 24
            FF(K)=0
            GO TO 25
 24         FF(K)=1
            BC(I)=BC(I)-1.0
 25         BC(I)=BC(I)*2.0
 23       CONTINUE
C******* END OF LOOP 23***************
          DO 31 K=1,A
            MM(K)=0
            MM(W)=1
```

```
 31        CONTINUE
           IF(FF(AA).EQ.1)GO TO 42
           IF(FF(AA).EQ.0)GO TO 37
C********* THE LOOP 121 USED TO FIND THE ROUNDED**************
C                  NUMBER STORED IN FINITE REGISTER
 42        NNN=AA
           DO 121 JJ=3,NNN
             II=NNN-JJ+2
             NN(II)=FF(II)+MM(II)+SS(II)
             IF(NN(II).LT.2)GO TO 121
             NN(II)=NN(II)-2
             SS(II-1)=1
 121       CONTINUE
C**********END OF LOOP 121************************
           GO TO 9
 37        DO 47 K=2,W
 47          NN(K)=FF(K)
 9         IF(FF(1).EQ.MM(1))GO TO 45
           NN(1)=1
           GO TO 41
 45        IF(FF(1).EQ.1)GO TO 6
           NN(1)=0
           GO TO 41
 6         NN(1)=1
 41        BA(I)=0.0
C********* THE LOOP 130 IS USED TO CONVERT THE ROUNDED**********
C                  BINERY NUMBER INTO THE DECIMEL NUMBER.
           DO 130 K=2,W
 130         BA(I)=BA(I)+NN(K)*(2.0**(-K+1))
C******END OF LOOP 130***************
           IF(NN(1).EQ.1)GO TO 131
           BD(I)=BA(I)
           GO TO 132
 131       BD(I)=-BA(I)
 132       DD(I)=B1(I)-BD(I)
 26      CONTINUE
C******* THIS PART OF THE PROGRAM IS USED TO STORE***********
C                  THE INFORMATION ABOUT THE ROUNDING
C                  OPTION.
         WRITE(10,300)W
         WRITE(1,600)W
         WRITE(1,600)(M+1)
         WRITE(2,600)(M+1)
         WRITE(10,301)
         WRITE(10,302)
         WRITE(10,303)
 600     FORMAT(20X,I5)
 300     FORMAT(4X,"WORD LENGTH :  ",I4)
 301     FORMAT(4X,"USED QUANTIZATION TYPE IS ROUNDING")
 302     FORMAT(4X,"I",3X,"COEFFICIENT B(I)",9X,"SCALED COEFFICIENT"
     1          ,5X,"ROUNDOFF ERROR")
 303     FORMAT(4X,"-",3X,"---------------",9X,"------------------"
     1          ,5X,"--------------")
         DO 304 I=1,M+1
```

```
      WRITE(10,305)I,B(I),B1(I),DD(I)
      WRITE(2,901)I,B(I),B1(I),DD(I)
304   CONTINUE
305   FORMAT(1X,I4,2X,E14.7,14X,E14.7,6X,E14.7)
      WRITE(10,306)
306   FORMAT(1X,"ROUNDED COEFFICIENT IN BINARY")
      DO 307 L=1,AA
        HH(L)=0
        FF(L)=0
        NN(L)=0
        SS(L)=0
        MM(L)=0
307   CONTINUE
      DO 331 I=1,M+1
        IF(B1(I).LT.(0.0)) GO TO 308
        FF(1)=0
        GO TO 309
308     FF(1)=1
309     BC(I)=2.0*ABS(B1(I))
        DO 310 K=2,AA
          IF(BC(I).GE.1.0)GO TO 311
          FF(K)=0
          GO TO 312
311       FF(K)=1
          BC(I)=BC(I)-1.0
312       BC(I)=2.0*BC(I)
310     CONTINUE
        DO 313 K=1,A
          MM(K)=0
          MM(W)=1
313     CONTINUE
        IF(FF(AA).EQ.1)GO TO 314
        IF(FF(AA).EQ.0)GO TO 315
314     NNN=AA
        DO 316 JJ=3,NNN
          II=NNN-JJ+2
          NN(II)=FF(II)+MM(II)+SS(II)
          IF(NN(II).LT.2)GO TO 317
          NN(II)=NN(II)-2
          SS(II-1)=1
          GO TO 316
317       NN(II)=NN(II)
316     CONTINUE
        GO TO 320
315     DO 321 K=2,W
321       NN(K)=FF(K)
320     IF(FF(1).EQ.MM(1))GO TO 322
        NN(1)=1
        GO TO 325
322     IF(FF(1).EQ.1)GO TO 324
        NN(1)=0
        GO TO 325
324     NN(1)=1
```

```
325       WRITE(10,326)(NN(L),L=1,W)
          WRITE(1,326)(NN(L),L=1,W)
326       FORMAT(12X,70(I1))
331     CONTINUE
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
        TYPE "IF YOU WANT SINUSOIDAL INPUT TYPE : HA "
        CALL CLOSE(2,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
 55     CONTINUE
C
C       END OF ROUNDING OPTION
C************************************************************
        RETURN
        END
```

USER'S MANUAL PROGRAM NEWC

FILE:	NEWC
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Finding the new filter coefficient.
FUNCTION:	This program is used to find the real filter coefficient values after they are changed in binary for nested filter structure.
PROGRAM USE:	The program is loaded by the following command:
	RLDR NEWC @FLIB@
SUBROUTINE REQUIRED:	None

FLOWGRAPH:

Type	Figure
1. Two's Complement of Binary Numbers	26
2. Binary to Decimal Number Converter	27

EXECUTION OF THE PROGRAM AND ITS RESULTS FOLLOW:

```
NEWC
ENTER THE OLD BINARY COEFFICIENT FILE NAME:  TC
ENTER THE NEW BINARY COEFFICIENT FILE NAME:  NTC
```

Content of the file TC is explained in the user's manual of our program IN.FR. The content of the file NTC is the same as the file FC which is also explained in the user's manual of the program in.FR.

```
C     ***      ****************************************      ******************
C
C                             NEW
C                             HARUN IHSAN
C                             OCTOBER 84
C     LANGUAGE:               FORTRAN 5
C
C     FUNCTION:               THIS PROGRAM IS USED TO  CALCULATE THE NESTED
C                             FILTER COEFFICIENT FROM  A GIVEN BINARY EQUIVI
C
C*****      **************************************   **********************
      DIMENSION YT(500)
      INTEGER OUTFILE(7),OPT
      INTEGER Y(20,140),OUTF(5)
      INTEGER W,OW,S,RR,G
      TYPE "ENTER THE OLD BINERY COEFFICIEN   LE NAME : "
      READ(11,10)OUTFILE(1)
 10   FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      READ(1,30)OW
      READ(1,30)S
 30   FORMAT(20X,I5)
 50   FORMAT(12X,70I1)
      ACCEPT"ENTER THE NEW BINERY COEFFICIENT   E NAME : "
      READ(11,400)OUTF(1)
 400  FORMAT(S15)
      CALL DFILW(OUTF,IER)
      IF(IER.EQ.13)GO TO 910
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
      CALL CFILW(OUTF,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      DO 40 I=0,(S-1)
        READ(1,50,END=41)(Y(I,K),K=1,OW)
 40   CONTINUE
C*****    ***************************************      ***
C
C     NEW COEFFICIENT
C
      DO 43 I=0,(S-1)
        YT(I)=0.0
        DO 140 II=2,OW
 140      YT(I)=YT(I)+Y(I,II)*(2.0**(-II+1))
        IF(Y(I,1).EQ.0)GO TO 43
        YT(I)=-YT(I)
 43   CONTINUE
C
C     END OF 43
C
C*****    ********************************************      ******
```

```
C
C*******THIS PART OF TRUNCATION IS USED TO WRITE*    *****
C                THE INFORMATION OBTAINED ABOVE
C                TO THE FILE
        WRITE FREE(2) (S-1)
        WRITE FREE(2) 0
        DO 44 I=0,(S-1)
  44      WRITE FREE(2)YT(I)
        WRITE FREE(2) 1.
        WRITE FREE(2) 1.
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
        CALL CLOSE(2,IER)
        IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
        STOP
        END
```

USER'S MANUAL PROGRAM NES1

FILE:	NES1
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Finding the Nested Filter Coefficients.

FUNCTION: This program locates the nested filter coefficients based on the equation below:

BN(0) = A(0)

BN(I) = A(I)/QUANTIZED(A(I)))

where BN = nested structure coefficient; A = direct form coefficient; and QUANTIZED(A(I)) = truncated or rounded direct form coefficient.

Then, the nested filter coefficients are scaled such that each coefficient is two times less than the absolute maximum value of the coefficient. Finally, those coefficients are quantized according to user requirements of either the truncating or the rounding technique.

PROGRAM USE: The program is loaded by the following command:

RLDR NES1 @FL1B@

SUBROUTINE REQUIRED: None

FLOWGRAPH:

Type	Figure
1. Decimal to Binary Number Conversion	25

EXECUTION OF THE PROGRAM AND ITS RESULTS FOLLOW:

```
NES1
COEFFICIENT WORD LENGTH:  16
INPUT FILE NAME FOR NESTED STRUCTURE:  TC1
ENTER FILE NAME FOR NESTED COEFFICIENT:  NC
QUANTIZE TYPE (1-TRUNCATION, 0-ROUNDING) 1
```

The content of the file TC1 is explained in the user's manual of the program IN.FR. The file NC contains the coefficients number at the first and the nested coefficients (in binary) at the second column. The first number 6 represents the number of coefficients and the second number 16, the desired coefficient word length.

NC

```
      6
     16
 1            0000000000001001
 2            0100000000000000
 3            0001001100100110
 4            0000101111010001
 5            0000011101001011
 6            0000001000110101
```

```
C***      ****************************************      ****************
C
C        PROGRAM            NEST
C        AUTHOR             HARUN INANLI
C        DATE               SEPTEMBER 83
C        LANGUAGE           FORTRAN 5
C
C        FUNCTION           THIS PROGRAM CALCULATES    NESTED FILTER
C                           STRUCTURE COEFFICIENT L.   O UN THE EQUATION
C                           BELOW
C                           BN(0)=A(0)
C                           BN(I)=A(I)/QUANTIZED(A(
C        WHERE              BN : NESTED STRUCTURE C    ICIENT
C                           A  : THE SCALED DIRECT F   M COEFFICIENT
C
C                           THE SCALED DIRECT FORM C   FFICIENTS ARE FOUND BY
C                           THE PRONGRAM IN.FR  FURT   HORE THE NESTED FILTER
C                           COEFFICIENTS ARE SCALED    QUANTIZED  THE QUANT
C                           CAN BE DONE EITHER IN T    TION OR IN ROUND NG.
C
C***************************************************      ********************
      REAL BN
      INTEGER OUTFILE(7),S,I,OPT,NC
      INTEGER BB(20,140),SS(20,140),MM(20,140
      INTEGER NN(20,140)
      DIMENSION X(20),XS(20),D(20),XQ(20),BN(
      DIMENSION BX(20),BS(20)
      ACCEPT"COEFFICIENT WORD LENGTH : ",NC
      ACCEPT"INPUT FILE NAME FOR NESTED STRUC      : "
      READ(11,10)OUTFILE(1)
 10   FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      READ(1,20)S
 20   FORMAT(20X,I5)
      DO 30 I=1,S
 30     READ(1,40)I,X(I),XS(I),D(I)
 40   FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E14.7)
      CALL CLOSE(1,IER)
      IF(IER NE.1)TYPE"CLOSE FILE ERROR",IER
      ACCEPT"ENTER FILE NAME FOR NESTED COEFF      ENT : "
      READ(11,11)OUTFILE(1)
 11   FORMAT(S15)
      CALL DFILW(OUTFILE,IER)
      IF(IER.EQ.13)GO TO 999
      IF(IER NE 1)TYPE"DELETE FILE ERROR",IER
 999  CALL CFILW(OUTFILE,2,IER)
      IF(IER NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTFILE,3,IER)
      IF(IER NE 1)TYPE"OPEN FILE ERROR",IER
```

```
      ACCEPT"QUANTIZATION TYPE (1-TRUNCATION, 0-ROUNDING) ",OPT
      DO 41 I=1,S
        BS(I)=0.0
        BN(I)=0.0
        XS(I)=0.0
        DO 42 II=1,NC
42        B(I,II)=0
41    CONTINUE
C*************************************************************
C
C        THIS PART IS USED TO FIND NESTED STRUCTUR
C                COEFFICIENT IN REAL
C
      BN(1)=B(1)
      DO 50 I=2,S
50      BN(I)=B(I)/XS(I-1)
C
C        NESTED STRUCTURE COEFFICIENT
C
C*************************************************************
C
C*************************************************************
C
C        THIS PART IS USED TO SCALE THE NESTED STRUCTURE
C                COEFFICIENT
C
      BM=0.0
      DO 51 I=1,S
        IF(ABS(BN(I)).LT.BM)GO TO 51
        BM=ABS(BN(I))
51    CONTINUE
      DO 54 I=1,S
54      BS(I)=0.0
      WRITE(2,131)S
      WRITE(2,131)NC
      DO 52 I=1,S
        BS(I)=BN(I)/(2*BM)
52    CONTINUE
C
C
C
C*************************************************************
```

```
C
C*****  *****************************************************      ****
C
C         THIS PART IS USED TO CONVERT THE REAL           ENT
C                 INTO THE BINARY
C
          DO    I=1,N
           IF (BS(I).LT.0.0) GO TO 80
           BB(I,1)=0
           GO TO 90
   80      BB(I,1)=1
   90      BX(I)=2.0*ABS(BS(I))
           DO 150 II=2,NC+1
             IF(BX(I).GE.1.0)GO TO 110
             BB(I,II)=0
             GO TO 120
  110        BB(I,II)=1
             BX(I)=BX(I)-1.0
  120        BX(I)=2.0*BX(I)
  150      CONTINUE
C
C         BINARY
C
C*****  *****************************************************      *****
C
C*****  *****************************************************      *****
C
C         THIS PART IS USED FOR STORING THE TRUNCAT
C                 NESTED STRUCTURE COEFFICIENT NUM
C
           IF(OPT.EQ.0)GO TO 50
  170      FORMAT(5X,I4)
           WRITE(10,130)(I,(BB(I,II),II=1,NC))
           WRITE(2,130)(I,(BB(I,II),II=1,NC))
  130      FORMAT(1X,I4,10X,140(I1))
           GO TO 151
C
C         TRUNCATION
C
C*****************************************************************    *****
C
```

```
C*****      ******************************************      ********
C
C          [illegible] PART IS USED TO STORE THE ROUNDED
C                NESTED STRUCTURE COEFFICIENT NUM
C
   [illegible]   IF(BB(I,(NC+1)).EQ.O)GO TO 160
         DO 180 II=1,(NC-1)
  180      MM(I,II)=0
         MM(I,NC)=1
         DO 190 II=1,NC
           SS(I,II)=0
           MM(I,II)=0
  190    CONTINUE
         DO [illegible] II=2,NC
           [illegible]
           [illegible] BB(I,J)
           [illegible] TO 200
           [illegible] (I,J
           [illegible]
  [illegible]  CONTINUE
         [illegible] BB [illegible]
         GO TO 500
  [illegible]  DO [illegible] II=1,NC
  [illegible]    NN(I,II)=BB(I,II)
  500    WRITE(10,130)(I,(NN(I,II),II=1,NC))
         WRITE(2,130)(I,(NN(I,II),II=1,NC))
  151    CONTINUE
C
C          ROUND NO
C
C*****      ******************************************
         STOP
         END
```

USER'S MANUAL PROGRAM HA

FILE:	HA
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Creating the input.
FUNCTION:	This program produces the input, according to user requirements, in sinusoidal, step or multiple-step function and then scales it. Finally, it quantizes the input function, according to user requirements, either by the truncating or rounding technique.
PROGRAM USE:	The program is loaded by the following command:

RLDR HA HA1 STEP MSTE HA2 HA3 @FLIB@

SUBROUTINE REQUIRED:

Name	Location	Purpose
HA1	DP4:OWEN	To produce sinusoidal function
STEP	DP4:OWEN	To produce step function
MSTE	DP4:OWEN	To produce multiple-step function
HA2	DP4:OWEN	To scale the input
HA3	DP4:OWEN	To quantize the input

EXECUTION OF THE PROGRAM AND ITS RESULTS FOLLOW:

```
HA
ENTER FILE NAME:  TI
NUMBER OF SAMPLES:  10
INPUT TYPE (1-STEP, 0-SINUSOIDAL) 1
AMOUNT OF STEP:  5
WORD LENGTH:  16
ENTER FILE NAME FOR INPUT:  TI1
QUANTIZATION TYPE (1-TRUNCATION, 0-ROUNDING) 1
```

File TI shown below, containes the desired number of samples with 10, coefficient word length with 16, and the coefficients in binary. The content of the file TI1 is the same as the file TC1 explained in the user's manual of program IN.FR.

TI

```
10
16
0000110011001100
0000110011001100
0000110011001100
0000110011001100
0000110011001100
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
```

```
C ****************************************************   *****************
C
C        PROGRAM           HA
C        AUTHOR   :        HARUN INANLI
C        DATE     :        SEPTEMBER 83
C        LANGUAGE:         FORTRAN 5
C
C        FUNCTION:         THIS PR OGRAM PRODUC  , STEP,MULTIPLE STEP
C                          OR SINUSOIDAL INPUT   CORDING TO USER
C                          REQUIREMENT. THEN QU  TIZE THE INPUT EITHER
C                          IN TRUNCATING OR IN   UNDING TECHNIQUE.
C
C*******************************************   **********************
        DIMENSION X(500),XX(500),XS(500),BN(   ),BK(500)
        DIMENSION BB(256),D(256),BE(256),BD(   ),DD(256)
        DIMENSION BA(500)
        REAL TI
        INTEGER   A,HH(70),K,MM(70),NN(70),F   70),OPT
        INTEGER SS(70),OUTFILE(7),RA,MRA
        ACCEPT "ENTER FILE NAME : "
        READ(11,11)OUTFILE(1)
 11     FORMAT(S  )
        CALL DFILW(OUTFILE,IER)
        IF(IER.EQ.13)GO TO  906
        IF(IER.NE.1)TYPE"DELETE FILE ERROR",  R
 906    CALL CFILW(OUTFILE,2,IER)
        IF(IER.NE.1)TYPE"CREATE FILE ERROR",  R
        CALL OPEN(2,OUTFILE,3,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IE
        ACCEPT"NUMBER OF SAMPLES :  " ,R
        ACCEPT"INPUT TYPE(0-STEP,1-MSTEP,2-S  USOIDAL)",OPT1
        DO 10 L=1,R
 10       X(L)=0.0
        IF(OPT1.EQ.2)GO TO 100
        IF(OPT1.EQ.1)GO TO 103
        IF(OPT1.EQ.0)GO TO 102
 100    CALL HA1(X,R)
        GO TO 101
 103    CALL MSTE(X,R,RA,MRA)
        GO TO 101
 102    CALL STEP(X,R,RA)
 101    CALL HA2(X,XS,K,R)
        CALL HA3(X,XS,K,R)
        CALL CLOSE(2,IER)
        IF(IER.NE.1)TYPE "CLOSE FILE ERROR",  R
        STOP
        END
```

USER'S MANUAL SUBROUTINE HA1

FILE:	HA1
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Producing Sinusoidal Function.
FUNCTION:	This program produces the sinusoidal function according to the equation below:

X(N) = TT*Sin(N*2* /T) + 1.0

where TT = gain
N = number of points up to 500
T = period

By inspection of this equation, the sinusoidal function values will be all positive.

SUBROUTINE REQUIRED: None

USER'S MANUAL SUBROUTINE STEP

FILE:	STEP
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Producing Step Function.

FUNCTION: This subroutine produces the step function up to 500 points. The magnitude of step function is 0.1.

SUBROUTINE REQUIRED: None

USER'S MANUAL SUBROUTINE HA2

FILE: HA2

DIRECTORY: DP4:OWEN

LANGUAGE: FORTRAN 5

DATE: September 1983

AUTHOR: Harun Inanli

SUBJECT: Scaling the Input Function.

FUNCTION: This subroutine scales the input signal such that the absolute maximum value of the signal is less than 0.1.

SUBROUTINE REQUIRED: None

USER'S MANUAL SUBROUTINE MSTE

FILE: MSTE

DIRECTORY: DP4:OWEN

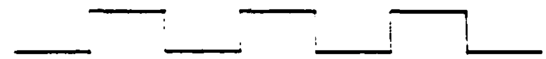
LANGUAGE: FORTRAN 5

DATE: September 1983

AUTHOR: Harun Inanli

SUBJECT: Producing the Multiple Step Function.

FUNCTION: This subroutine produces the step function as shown below.



The magnitude of the step is 0.1.

SUBROUTINE REQUIRED: None

USER'S MANUAL SUBROUTINE HA3

FILE: HA3

DIRECTORY: DP4:OWEN

LANGUAGE: FORTRAN 5

DATE: September 1983

AUTHOR: Harun Inanli

SUBJECT: Quantizing the Input Signal.

FUNCTION: This subroutine quantizes the scaled input signal according to user requirements of either the truncating or rounding technique. First, scaled input is converted into the binary and placed in the input register. The input register can be a maximum 140 bits long. Then, according to user requirement, this binary number is truncated or rounded to the desired finite word length. Finally, quantized number is converted back to real number and stored in the file.

SUBROUTINE REQUIRED: None

FLOWGRAPH:

Type	Figure
1. Decimal to Binary Number Converter	25
2. Two's Complement of Binary Numbers	26
3. Binary to Decimal Number Conversion	27

```
C*****************************************************************************
C
C         PROGRAM           HA1
C         AUTHOR            HARUN INANLI
C         DATE              SEPTEMBER 83
C         LANGUAGE          FORTRAN 5
C
C         FUNCTION :        THE SUBROUTINE IS USE  TO PRODUCE A
C                           SINUSOIDAL SIGNAL FOR  NPUT ACCORDING TO
C                           USER REQUARMENT.
C
C*****************************************************************************
          SUBROUTINE HA1(X,R)
          DIMENSION X(500)
          REAL TT,T
          INTEGER R
          ACCEPT "WHAT IS THE PERIOD :  ",T
          ACCEPT "WHAT IS THE GAIN :  ",TT
          DO 10 N=1,R
   10       X(N)=TT*SIN((FLOAT(N)*2*3.14159)/T  )0
          RETURN
          END
```

```
C*****************************************************************************
C
C         PROGRAM :         STEP
C         AUTHOR  :         HARUN INANLI
C         DATE    :         SEPTEMBER 83
C         LANGUAGE:         FORTRAN 5
C
C         FUNCTION:         THIS SUBROUTINE IS  SED TO PRODUCE
C                           THE STEP INPUT.
C
C*****************************************************************************
          SUBROUTINE STEP(X,R,RA)
          DIMENSION X(500)
          INTEGER RA,R
          ACCEPT"AMOUNT OF STEP",RA
          DO 10 I=0,RA
   10       X(I)=1
          DO 20 I=(RA+1),R-1
   20       X(I)=0.0
          RETURN
          END
```

```
C*************************************************************
C
C        PROGRAM  :        HA2
C        AUTHOR   :        HARUN INANLI
C        DATE     :        SEPTEMBER 83
C        LANGUAGE :        FORTRAN 5
C
C        FUNCTION :        SUBROUTINE HA2 IS USED THE SCALE THE
C                          PRUDUCED INPUT SIGNAL SUCH THAT THE
C                          MAXIMUM VALUE OF THE SIGNAL LESS THAN
C                          .1
C*****************************************************************
C
        SUBROUTINE HA2(X,XS,K,R)
        DIMENSION XX(500),XS(500),X(500)
        INTEGER R,K
        REAL XXM,L
        XXM=0.0
C******THE LOOP 10 USED TO FIND THE MAXIMUM VALUE*********
C
        DO 10 N=1,R
          XX(N)=ABS(X(N))
          IF(XX(N).GE.XXM)GO TO 20
          XS(N)=X(N)
          GO TO 10
  20      XXM=XX(N)
          XS(N)=X(N)
  10    CONTINUE
C
C******END OF LOOP 10***************
        L=XXM/.1
C*******THE LOOP 30 IS USED TO SCALE THE INPUT**********
C
        DO 30 I=1,R
  30      XS(I)=XS(I)/FLOAT(L)
C
C*****END OF LOOP 30*************
        RETURN
        END
```

```
C************** ************************* *********
C
C       PROGRAM :        MSTE
C       AUTHOR  :        HARUN INANLI
C       DATE    :        SEPTEMBER 83
C       LANGUAGE:        FORTRAN 5
C
C       FUNCTION.        THIS SUBROUTINE IS USED TO PRODUCE
C                        THE MULTIPLE STEP INPUT.
C
C***************************************** ***************
        SUBROUTINE MSTE(X,R,RA,MRA)
        DIMENSION X(500)
        INTEGER RA,MRA,R
        ACCEPT"AMOUNT OF STEP : ",RA
        MRA=0
 21     IF(I.GE.R)GO TO 22
        DO 10 I=MRA,(RA+MRA)
 10       X(I)=.1
        MRA=I
        DO 20 I=MRA,(MRA+RA)
 20       X(I)=0.0
        MRA=I
        IF(I.LT.R)GO TO 21
 22     RETURN
        END
```

```
C*********************************************************************
C
C         PROGRAM   :       HA3
C         AUTHOR    :       HARUN INANLI
C         DATE      :       SEPTEMBER 83
C         LANGUAGE  :       FORTRAN 5
C
C         FUNCTION  :       SUBROUTINE HA3 IS USED TO QUANTIZE THE
C                           SCALED INPUT EITHER IN TRUNCATED OR ROUNDING
C                           TECHNIQUE ACCORDING TO USER REQIEREMENT.  THEN
C                           CALCULATE THE QUANTIZATION ERROR AND STORE ALL
C                           THESE INFORMATION IN THE FILE
C
C*********************************************************************
          SUBROUTINE HA3(X,XS,K,R)
          DIMENSION X(500),XS(500),BN(500),BB(500)
          DIMENSION BK(500),BA(500),BD(500),DD(500),D(500),BE(500)
          INTEGER HH(500),K,MM(70),NN(70),FF(70),OPT,SS(70)
          INTEGER A,R,AA,OUTF(5)
          ACCEPT"WORD LENGTH : ",K
          ACCEPT"ENTER FILE NAME FOR INPUT : "
          READ(11,900)OUTF(1)
   900    FORMAT(S15)
          CALL DFILW(OUTF,IER)
          IF(IER.EQ.13)GO TO 910
          IF(IER.NE.1)TYPE"DELETE FILE ERROR)",IER
   910    CALL CFILW(OUTF,2,IER)
          IF(IER.NE.1) TYPE "CREATE FILE ERROR",IER
          CALL OPEN(1,OUTF,3,IER)
          IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
          ACCEPT"QUANTIZATION TYPE (1-TRUNCATION,0-ROUNDING)",OPT
          A=K-1
          A1=A-1
          AA=K+1
          DO 56 L=1,K
            HH(L)=0
            FF(L)=0
            NN(L)=0
            SS(L)=0
            MM(L)=0
   56     CONTINUE
          IF(OPT.EQ.1)GO TO 11
          IF(OPT.EQ.0)GO TO 91
C***************************************************************
C
C         TRUNCATION OPTION
C
   11     DO 10 I=1,R
            IF(XS(I).LT.0.0)GO TO 81
            HH(1)=0
            GO TO 82
   81       HH(1)=1
```

```
82        BB(I)=2.0*ABS(XS(I))
C********THE LOOP 20 IS USED TO CONVERT THE *************
C                  DECCIMEL NUMBER TO BINERY.
          DO 20 N=2,K
            IF(BB(I).GE.1.0)GO TO 30
            HH(N)=0
            GO TO 40
30          HH(N)=1
            BB(I)=BB(I)-1.0
40          BB(I)=BB(I)*2.0
20        CONTINUE
C******END OF LOOP 20****************************
          BK(I)=0.0
C************THE LOOP 60 IS USED TO CONVERT THE **********
C                  BINERY NUMBER TO DECIMEL.
          DO 60 N=2,K
60          BK(I)=BK(I)+HH(N)*(2.0**(-N+1))
C*********END LOOP 60*****************
          IF(HH(1).EQ.1)GO TO 100
          BN(I)=BK(I)
          GO TO 110
100       BN(I)=-BK(I)
110       D(I)=XS(I)-BN(I)
10      CONTINUE
C*******THE INFORMATION OPTAINED ABOVE IS STORED IN THE FILE**********
        WRITE(10,204)R
        WRITE(10,205)K
        WRITE(2,400)R
        WRITE(1,400)R
        WRITE(2,400)K
        WRITE(10,206)
        WRITE(10,200)
        WRITE(10,201)
400     FORMAT(20X,I5)
204     FORMAT(4X,"NUMBER OF SAMPLE :  ",I9)
205     FORMAT(4X,"WORD LENGTH :       ",I9)
206     FORMAT(4X,"USED QUANTIZATION TYPE IS TRUNCATION")
200     FORMAT(4X,"I",6X,"INPUT X(I)",5X,"SCALED XS(I)",2X,"ROUNDOFF ERR
201     FORMAT(4X,"-",6X,"----------",4X,"------------",2X,"-----------
        DO 203 I=1,R
          WRITE(10,202)I,X(I),XS(I),D(I)
          WRITE(1,202)I,X(I),XS(I),D(I)
203     CONTINUE
202     FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E14.7)
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
        WRITE(10,210)
210     FORMAT(1X,"TRUNCATED INPUT IN BINARY")
        DO 207 I=1,R
          IF(XS(I).LT.0.0)GO TO 212
          HH(1)=0
          GO TO 213
212       HH(1)=1
```

```
  213     BB(I)=2.0*ABS(XS(I))
          DO 214 N=2,K
            IF(BB(I).GE.1.0)GO TO 215
            HH(N)=0
            GO TO 216
  215       HH(N)=1
            BB(I)=BB(I)-1.0
  216       BB(I)=BB(I)*2.0
  214     CONTINUE
          WRITE(2,208)(HH(N),N=1,K)
          WRITE(10,208)(HH(N),N=1,K)
  208     FORMAT(12X,200(I1))
  207   CONTINUE
        GO TO 55
C
C       END OF TRUNCATION OPTION
C
C*********************************************
C**************************************************
C
C       ROUNDING OPTIION
C
  91    DO 26 I=1,R
          FF(1)=0
          IF(XS(I).LT.(0.0))GO TO 21
          FF(1)=0
          GO TO 22
  21      FF(1)=1
  22      BE(I)=2.0*ABS(XS(I))
C******THE LOOP 23 IS USED TO CONVERT THE *************
C                 DECIMEL NUMBER TO BINERY
          DO 23 N=2,AA
            IF(BE(I).GE.1.0)GO TO 24
            FF(N)=0
            GO TO 25
  24        FF(N)=1
            BE(I)=BE(I)-1.0
  25        BE(I)=BE(I)*2.0
  23      CONTINUE
C**********END OF LOOP 23*******************************
          DO 31 N=1,K
            MM(N)=0
            MM(K)=1
  31      CONTINUE
          IF(FF(AA).EQ.1)GO TO 42
          IF(FF(AA).EQ.0)GO TO 37
  42      NNN=AA
C************THE LOOP 121 IS USED TO FIND ROUNDED***********
C                NUMBER STORED IN FINITE REGISTER
          DO 121 JJ=3,NNN
            II=NNN-JJ+2
            NN(II)=FF(II)+MM(II)+SS(II)
            IF(NN(II).LT.2)GO TO 121
            NN(II)=NN(II)-2
            SS(II-1)=1
  121     CONTINUE
```

```
C*********END OF LOOP 121*******************************
      . GO TO 9
 37     DO 47 N=2,K
          NN(N)=FF(N)
 47     CONTINUE
 9      IF(FF(1).EQ.MM(1))GO TO 45
        NN(1)=1
        GO TO 41
 45     IF(FF(1).EQ.1)GO TO 6
        NN(1)=0
        GO TO 41
 6      NN(1)=1
 41     BA(I)=0.0
C*******THE LOOP 130 IS USED TO CONVERT THE ROUNDED ***********
C                BINERY NUMBER INTO THE DECIMEL NUMBER
        DO 130 N=2,K
 130      BA(I)=BA(I)+NN(N)*(2.0**(-N+1))
C********END OF LOOP 130*************************
        IF(NN(1).EQ.1)GO TO 131
        BD(I)=BA(I)
        GO TO 132
 131    BD(I)=-BA(I)
 132    DD(I)=XS(I)-BD(I)
 26    CONTINUE
C*************THIS PART OF THE PROGRAM IS USED TO STORE***********
C                THE INFORMATION ABOUT THE ROUNDING OPTION
       WRITE(10,300)R
       WRITE(10,301)K
       WRITE(2,400)R
       WRITE(1,400)R
       WRITE(2,400)K
       WRITE(10,302)
       WRITE(10,303)
       WRITE(10,304)
       DO 305 I=1,R
         WRITE(10,306)I,X(I),XS(I),DD(I)
         WRITE(1,306)I,X(I),XS(I),DD(I)
 305   CONTINUE
 306   FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E14.7)
       DO 340 L=1,K
         HH(L)=0
         FF(L)=0
         NN(L)=0
         SS(L)=0
         MM(L)=0
 340   CONTINUE
       WRITE(10,341)
 341   FORMAT(1X,"ROUNDED INPUT IN BINARY")
       DO 310 I=1,R
         IF(XS(I).LT.(0.0))GO TO 311
         FF(1)=0
         GO TO 312
 311     FF(1)=1
 312     BE(I)=2.0*ABS(XS(I))
```

```
      DO 313 N=2,AA
        IF(BE(I).GE.1.0)GO TO 314
        FF(N)=0
        GO TO 315
314     FF(N)=1
        BE(I)=BE(I)-1.0
315     BE(I)=2.0*BE(I)
313   CONTINUE
      DO 317 N=1,K
        MM(N)=0
        MM(K)=1
317   CONTINUE
      IF(FF(AA).EQ.1)GO TO 318
      IF(FF(AA).EQ.0)GO TO 319
318   NNN=AA
      DO 320 JJ=3,NNN
        II=NNN-JJ+2
        NN(II)=FF(II)+MM(II)+SS(II)
        IF(NN(II).LT.2)GO TO 321
        NN(II)=NN(II)-2
        SS(II-1)=1
        GO TO 320
321     NN(II)=NN(II)
320   CONTINUE
      GO TO 322
319   DO 326 N=2,K
326     NN(N)=FF(N)
322   IF(FF(1).EQ.MM(1))GO TO 327
      NN(1)=1
      GO TO 331
327   IF(FF(1).EQ.1)GO TO 330
      NN(1)=0
      GO TO 331
330   NN(1)=1
331   WRITE(2,332)(NN(L),L=1,K)
      WRITE(10,332)(NN(L),L=1,K)
332   FORMAT(12X,200(I1))
310   CONTINUE
300   FORMAT(4X,"NUMBER OF SAMPLES :  ",I9)
301   FORMAT(4X,"WORD LENGTH       :  ",I9)
302   FORMAT(4X,"USED QUANTIZATION TYPE IS ROUNDING")
303   FORMAT(4X,"I",6X,"INPUT X(I)",5X,"SCALED XS(I)",2X,"ROUNDOFF ERR
304   FORMAT(4X,"-",6X,"----------",5X,"------------",2X,"-----------
55    CONTINUE
      TYPE "IF YOU WANT OUTPUT TYPE :OUT "
C
C     END OF ROUNDING OPTION
C
C***********************************************************
      RETURN
      END
```

Appendix C

Digital Filter Structure

Appendix C contains the program and user's manual for different digital filter structures. Each program user's manual explains what the program does. These are called as follows:

1. OUT
2. COUT
3. POUT
4. NES
5. CNES
6. PNES

USER'S MANUAL PROGRAM OUT

FILE: TOUT

DIRECTORY: DP4:OWEN

LANGUAGE: FORTRAN 5

DATE: September 1983

AUTHOR: Harun Inanli

SUBJECT: Calculating the Direct Form Digital Filter Response.

FUNCTION: This program is used to compute the direct form digital filter output response. The digital filter coefficient and input signal are taken from two different files in binary. Then, they are multiplied and added based on convolution. The addition is carried out in two's complement. The output register is two times larger than the input register and the output response is stored in binary.

PROGRAM USE: The program is loaded by the following command:

```
RLDR TOUT @FLIB@
```

SUBROUTINE REQUIRED: None

FLOWGRAPH:

Type		Figure
1.	Two's Complement of Binary Numbers	26
2.	Two's Coplement Addition	28
3.	Binary Multiplication	29
4.	Shift-left and Shift-right Operator	30
5.	FIR Direct Form Structure	31

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
TOUT
BINARY COEFFICIENT FILE NAME:  TC
BINARY INPUT FILE NAME:  TI
UNQUANTIZE BINARY OUTPUT NAME:  TO
```

The content of the file TC and TI is explained in Appendix B. The file TO shown below contains the desired word length with 16, number of samples with 10, and the output response in binary.

<u>TO</u>

```
        16
        10
0    000000000001110111101101000100000
1    000000001011010100010111101100000
2    000000011001010100110011010000000
3    000000100111010101001110110100000
4    000000110000110001111001011100000
5    000000110000110001111001011100000
6    000000100111010101001110110100000
7    000000011001010100110011010000000
8    000000001011010100010111101100000
9    000000000001110111101101000100000
```

```
C*********************************************************************
C
C       PROGRAM :        OUT
C       AUTHOR  :        HARUN INANLI
C       DATE    :        SEPTEMBER 83
C       LANGUAGE:        FORTRAN 5
C
C       FUNCTION:        THIS PROGRAM IS USED TO FIND THE FILTER
C                        OUTPUT BASED ON CONVOLUTION. THE BINERY INPUT
C                        AND FILTER COEFFICICENT ARE COMING FROM THE FILES
C                        THESE VALUES ARE CALCULATED BY PROGRAM HA AND IN.
C                        RESPECTIVELY. NEGATIVE NUMBER IS CONVERTED TO THE
C                        TWO'S COMPLEMENT THEN ADDITION IS CARIED OUT IN
C                        THIS NUMBER SYSTEM. THE OUTPUT WORD LENGTH IS
C                        SPECIFIED TWO TIMES BIGGER THAN INPUT WORD LENGTH
C                        THE CALCULATED OUTPUTS ARE STORE IN BINERY IN THE
C                        FILE
C
C**************************************************************************

      INTEGER OUTFILE(7),OUTF(7)
      INTEGER X(20,140),H(20,140),PP(20,140),YC(20,140)
      INTEGER P(20,140),SS(20,140),YY(20,140)
      INTEGER IW,NC,CW,S,F,RR,R2,V,JB,JA
      ACCEPT"BINERY COEFFICIEN FILE NAME : "
      READ(11,50)OUTFILE(1)
50    FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      READ(1,60)CW
60    FORMAT(20X,I5)
      READ(1,60)NC
      DO 70 I=0,(NC-1)
70      READ(1,80)(H(I,K),K=1,CW)
80    FORMAT(12X,140(I1))
      CALL CLOSE(1,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
      ACCEPT"BINERY INPUT FILE NAME : "
      READ(11,10)OUTFILE(1)
10    FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN INPUT FILE ERROR : ",IER
      READ(1,30) S
30    FORMAT(20X,I5)
      READ(1,30)IW
      ACCEPT "UNQUANTIZED BINERY OUTPUT NAME : "
      READ(11,905)OUTF(1)
905   FORMAT(S15)
      CALL DFILW(OUTF,IER)
      IF(IER.EQ.13)GO TO 906
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
906   CALL CFILW(OUTF,2,IER)
```

```
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      WW=2*IW
      WWW=2*IW+1
      IWW=IW+1
      WW1=2*IW+2
      CWW=CW+1
      DO 400 I=0,(S-1)
        DO 410 K=IWW,WWW
          XA(I,K)=0
          X(I,K)=0
410     CONTINUE
        DO 401 M=0,(NC-1)
          IF(M.GT.I)GO TO 400
          DO 402 K=1,WWW+2
            SS(M,K)=0
402       CONTINUE
401     CONTINUE
400   CONTINUE
      DO 430 M=0,(NC-1)
        DO 440 K=CWW,WWW
440       H(M,K)=0
430   CONTINUE
      WRITE(2,915)IW
      WRITE(2,916)S
40    FORMAT(12X,140(I1))
      J=0
      RR=0
      JB=0
433   JB=J
      DO 435 J=JB,(JB+9)
        DO 436 K5=1,WW1
          YY(J,K5)=0
          YC(J,K5)=0
436     CONTINUE
435   CONTINUE
      IF(JB.EQ.297)GO TO 467
      IF(JB.EQ.198)GO TO 467
      IF(JB.EQ.99)GO TO 467
      TYPE(RR)
      IF(RR.EQ.400)GO TO 458
      IF(RR.EQ.300)GO TO 458
      IF(RR.EQ.200)GO TO 458
      IF(RR.EQ.100)GO TO 458
467   DO 20 JA=RR,(RR+9)
20      READ(1,40,END=41)(X(JA,K),K=1,IW)
```

```
C*****************************************************************
C
C        THE BEGINING OF THE CONVOLUTION
C
  41     CONTINUE
  458    RR=JA
         DO 921  J=JB, (JB+9)
           IF(J.GT.(S-1))GO TO 929
           DO 110 M=0,NC-1
             LL=J-M
             IF(LL.LT.0)GO TO 921 .
             IF(J.GE.(JB+9))GO TO 433
             DO 960 II=1,WWW+2
               P(M,II)=0
               SS(M,II)=0
 960         CONTINUE
C********THE LOOP 130 IS USED FOR BINERY MILTIPLICATION******
C
             DO 130 R=2,CW
               KK=CW-R+2
               IF(H(M,KK).EQ.1)GO TO 150
C*********THE LOOP 160 IS USED FOR SHIFT-RIGHT***********
C
  121          DO 160 K=2,WWW
                 K1=WWW-K+2
                 P(M,K1+1)=P(M,K1)
  160          CONTINUE
               P(M,2)=0
C
C**************END OF LOOP 160*****************
               GO TO 130
  150          DO 180 JJ=2,WWW
                 II=WWW-JJ+2
                 P(M,II)=X(LL,II)+P(M,II)+SS(M,II)
                 IF(P(M,II).LT.2)GO TO 180
                 P(M,II)=P(M,II)-2
                 SS(M,II-1)=1
  180          CONTINUE
               IF(SS(M,1).EQ.0)GO TO 121
  764          DO 528 K=2,WWW
                 K1=WWW-K+2
                 P(M,K1+1)=P(M,K1)
  528          CONTINUE
               P(M,2)=1
               GO TO 121
  130        CONTINUE
C
C*********END OF LOOP 130*************
```

```
          DO 190 II=2,WW
190         P(M,II)=P(M,II+1)
          IF(H(M,1).EQ.X(LL,1))GO TO 240
          P(M,1)=1
          GO TO 250
240       P(M,1)=0
C*******THE BEGINING OF THE ADDITION OF P AND YY*********
C
C
C***********THE BEGINING OF THE TWO'S COMPLIMENT OF P****
C
250       IF(P(M,1).EQ.0)GO TO 600
          DO 610 II=2,WWW
            IF(P(M,II).EQ.0)GO TO 620
            P(M,II)=0
            GO TO 610
620         P(M,II)=1
610       CONTINUE
          DO 602 II=1,WWW-1
            PP(M,II)=0
            SS(M,II)=0
602       CONTINUE
          PP(M,WWW)=1
          SS(M,WWW)=0
          DO 603 II=2,WWW
            JJ=WWW-II+2
            P(M,JJ)=P(M,JJ)+PP(M,JJ)+SS(M,JJ)
            IF(P(M,JJ).LT.2)GO TO 603
            P(M,JJ)=P(M,JJ)-2
            SS(M,JJ-1)=1
603       CONTINUE
600       DO 201 II=1,WWW
            JJ=WWW-II+1
            P(M,JJ+1)=P(M,JJ)
201       CONTINUE
          P(M,1)=0
C
C**********END OF THE TWO'S COMPLEMENT OF P*************
          DO 209 II=1,WW1
209         SS(M,II)=0
          DO 200 JJ=2,WW1
            II=WW1-JJ+1
            YY(J,II)=YY(J,II)+P(M,II)+SS(M,II)
            IF(YY(J,II).LT.2)GO TO 200
            YY(J,II)=YY(J,II)-2
            SS(M,II-1)=1
200       CONTINUE
C
C
C********END OF ADDITION OF P AND YY********************
```

```
            IF(SS(M,1).EQ.1)GO TO 781
            IF(SS(M,2).EQ.1)GO TO 781
            GO TO 184
  781       DO 678 II=1,WWW
              JJJ=WWW-II+1
              YY(J,JJJ+1)=YY(J,JJJ)
  678       CONTINUE
            YY(J,1)=0
  184       IF(M.EQ.(NC-1))GO TO 798
            IF(LL.EQ.0)GO TO 798
            IF(LL.GT.J)GO TO 929
            GO TO 110
C**********THE 183 IS USED FOR SHIFT-LEFT*********
C
  798       DO 183 II=1,WWW
  183         YC(J,II)=YY(J,II+1)
C
C********END OF LOOP 183**********************
            IF(YC(J,1).EQ.0)GO TO 800
            DO 810 II=2,WWW
              IF(YC(J,II).EQ.0)GO TO 820
              YC(J,II)=0
              GO TO 810
  820         YC(J,II)=1
  810       CONTINUE
            DO 819 II=1,WWW-1
              PP(J,II)=0
              SS(J,II)=0
  819       CONTINUE
            PP(J,WWW)=1
            SS(J,WWW)=0
            DO 829 II=2,WWW
              JJ=WWW-II+2
              YC(J,JJ)=YC(J,JJ)+PP(J,JJ)+SS(J,JJ)
              IF(YC(J,JJ).LT.2)GO TO 829
              YC(J,JJ)=YC(J,JJ)-2
              SS(J,JJ-1)=1
  829       CONTINUE
  800       CONTINUE
            WRITE(2,923)J,(YC(J,JJ),JJ=1,WWW)
  110     CONTINUE
  921   CONTINUE
C
C       END OF CONVOLUTION
C
C******************************************************************
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
  915   FORMAT(2X,I5)
  916   FORMAT(1X,I5)
  910   FORMAT(4X,"I",5X,"UNQUANTIZED OUTPUT")
  911   FORMAT(4X,"-",5X,"------------------")
  923   FORMAT(1X,I4,3X,140(I1))
        CALL CLOSE(2,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
  929   STOP
        END
```

USER'S MANUAL PROGRAM COUT

FILE:	COUT
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Calculating the Cascade Form of the Digital Filter Response.
FUNCTION:	This program computes the cascade form of the digital filter output response. Each second-order section coefficients and input signals are taken from two different files in binary. Then, for each second order, they are multiplied and added based on convolution. The addition is carried out in two's complement. The output of the first second-order section will be the input of the next second-order section. The final second order sec- tion output will be stored in the file as the cascade filter output.
PROGRAM USE:	The program is loaded by the following command: RLDR COUT @FLIB@
SUBROUTINE REQUIRED:	None

FLOWGRAPH:

Type		Figure
1.	Two's Complement of Binary Number	26
2.	Two's Complement Addition	28
3.	Binary Number Multiplication	29
4.	Shift-left and Shift-right Operator	31
5.	FIR Cascade Form Structure	32

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
COUT
BINARY COEFFICIENT FILE NAME:  TC
BINARY INPUT FILE NAME:  TI
UNQUANTIZE BINARY OUTPUT NAME:  TO
ENTER THE NEXT SECOND ORDER SECTION:  TO
NEXT SECOND ORDER OUTPUT FILE:  CTO
```

The content of the file TC and TI is explained in Appendix B. The file TO contains the output of the first second-order section output response in binary. The file CTO shown below, which contains the similar data explained for the file TO in Program OUT, represents the output response of the cascade form structure in binary.

TO

```
0    00000000011110111011010000000000
1    00000000111101011011100010100000
2    000000010111000101101100010100000
3    000000010111000101101100010100000
4    000000010111000101101100010100000
5    00000000111101011011100010100000
6    00000000011110111011010000000000
7    00000000000000000000000000000000
8    00000000000000000000000000000000
9    00000000000000000000000000000000
```

CTO

```
                16
                10
0    00000000000000110110001001101100 0
1    00000000000011100101000110111100 0
2    00000000001000001000100111000011 0
3    00000000001011101000111010101101 0
4    00000000001101011110010010111001 0
5    00000000001011101000111010101101 0
6    00000000001000001000100111000011 0
7    00000000000011100101000110111100 0
8    00000000000000110110001001101100 0
9    00000000000000000000000000000000
```

```
*********************************************************************

        PROGRAM :         CUUT
        AUTHOR  :         HARUN INANL1
        DATE    :         SEPTEMBER 83
        LANGUAGE:         FORTRAN 5

        FUNCTION:         THIS PROGRAM IS USED TO FIND THE FILTER
                          OUTPUT BASED ON CONVOLUTION BY USING THE CASCADE
                          FILTER STRUCTURE  THE NEGATIVE NUMBER IS
                          REPERESENTED IN TWO'S COMPLEMENT. THEN SUMMATION
                          IS CARRIED OUT IN THIS NUMBER SYSTEM, TOO.
                          THE OUTPUT VALUES IS STORED IN THE FILE.
                          EACH COMPONENT IS THE SECOND DEGREE FILTER

*********************************************************************

        INTEGER OUTFILE(7),OUTF(7),OUTD(7)
        INTEGER X(0:20,140),H(0:20,140),PP(0:20,140),YC(0:20,140)
        INTEGER P(0:20,140),SS(0:20,140),YY(0:20,140)
        INTEGER IW,NC,CW,S,F,RF,RR,JB,JA,QQ
        ACCEPT"BINERY COEFFICIEN FILE NAME : "
        READ(11,50)OUTFILE(1)
50      FORMAT(S15)
        CALL OPEN(1,OUTFILE,1,IER)
        READ(1,60)CW
60      FORMAT(20X,I5)
        READ(1,60)NC
        DO 70 I=0,(NC-1)
         READ(1,80)(H(I,K),K=1,CW)
70      CONTINUE
80      FORMAT(12X,140(I1))
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
        ACCEPT"BINERY INPUT FILE NAME : "
        READ(11,10)OUTFILE(1)
10      FORMAT(S15)
        CALL OPEN(1,OUTFILE,1,IER)
        IF(IER.NE.1)TYPE"OPEN INPUT FILE ERROR : ",IER
        READ(1,30) S
30      FORMAT(20X,I5)
        READ(1,30)IW
        ACCEPT "UNQUANTIZED BINERY OUTPUT NAME : "
        READ(11,905)OUTF(1)
905     FORMAT(S15)
        CALL DFILW(OUTF,IER)
        IF(IER.EQ.13)GO TO 906
        IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
906     CALL CFILW(OUTF,2,IER)
```

```
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      WW=2*IW
      WWW=2*IW+1
      IWW=IW+1
      WW1=2*IW+2
      CWW=CW+1
      DO 400 I=0,(S-1)
        DO 410 K=IWW,WWW
          X(I,K)=0
          XA(I,K)=0
410     CONTINUE
        DO 401 M=0,(NC-1)
          IF(M.GT.I)GO TO 400
          DO 402 K=1,WWW+2
            SS(M,K)=0
402       CONTINUE
401     CONTINUE
400   CONTINUE
      DO 430 M=0,(NC-1)
        DO 440 K=CWW,WWW
440       H(M,K)=0
430   CONTINUE
40    FORMAT(12X,140(I1))
C*****************************************************************
C
C     THE BEGINING OF CONVOLUTION FOR CASCADE FORM
C
      RF=0
412   J=0
      IF(RF.EQ.0)GO TO 513
      IF(RF.GT.(NC-3))GO TO 929
      ACCEPT"ENTER THE NEXT SECOND ORDER SECTION : "
      READ(11,905)OUTF(1)
      CALL OPEN(2,OUTF,1,IER)
      IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
      REWIND 2
      ACCEPT"NEXT SECOND ORDER OUTPUT FILE :"
      READ(11,10)OUTD(1)
      CALL DFILW(OUTD,IER)
      IF(IER.EQ.13)GO TO 584
      IF(IER.NE.1)TYPE "DELETE FILE ERROR",IER
584   CALL CFILW(OUTD,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(3,OUTD,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      IF(RF.NE.(NC-3))GO TO 513
      WRITE(3,915)IW
      WRITE(3,916)S
513   RR=0
      JB=0
```

```
C******THE BEGINING OF CONVOLUTION FOR SECOND*****************
C                 ORDER DIRECT FORM
 433    JB=J
        DO 435 J=JB,(JB+9)
          DO 436 K5=1,WW1
            YY(J,K5)=0
            YC(J,K5)=0
 436      CONTINUE
 435    CONTINUE
        IF(RF.EQ.0)GO TO 516
        IF(JB.EQ.297)GO TO 523
        IF(JB.EQ.198)GO TO 523
        IF(JB.EQ.99)GO TO 523
        TYPE RR
        IF(RR.EQ.400)GO TO 458
        IF(RR.EQ.300)GO TO 458
        IF(RR.EQ.200)GO TO 458
        IF(RR.EQ.100)GO TO 458
C******THE LOOP 21 IS USED TO READ THE OUTPUT OF THE******
C                 FIRST SECOND ORDER COMPONENT. THEN, IT
C                 IS USED AS INPUT FOR NEXT COMPONENT
C
 523    DO 21 JA=RR,(RR+9)
 21       READ(2,923,END=43,ERR=929)GQ,(X(JA,K),K=1,WWW)
 43     CONTINUE
C********END OF LOOP 21*******************
        GO TO 458
 516    IF(JB.EQ.297)GO TO 467
        IF(JB.EQ.198)GO TO 467
        IF(JB.EQ.99)GO TO 467
        TYPE RR
        IF(RR.EQ.400)GO TO 458
        IF(RR.EQ.300)GO TO 458
        IF(RR.EQ.200)GO TO 458
        IF(RR.EQ.100)GO TO 458
C**********THE LOOP 20 IS USED TO READ INPUT***********
C
 467    DO 20 JA=RR,(RR+9)
 20       READ(1,40,END=41,ERR=929)(X(JA,K),K=1,IW)
 41     CONTINUE
C
C**********END OF LOOP 20**********************
 458    RR=JA
        DO 921 J=JB,(JB+9)
          IF(RF.GT.(NC-1))GO TO 929
          IF(J.GT.(S-1))GO TO 932
          DO 110 M=0,2
            LL=J-M
            IF(LL.LT.0)GO TO 921
            IF(J.GE.(JB+9))GO TO 433
```

```
            DO 960 II=1,WWW+2
              P(M,II)=0
              SS(M,II)=0
  960       CONTINUE
C*********THE LOOP 130 IS USED FOR BINERY MILTIPLICATION********
C
            DO 130 R=2,CW
              KK=CW-R+2
              IF(H(M,KK).EQ.1)GO TO 150
  121         DO 160 K=2,WWW
                K1=WWW-K+2
                P(M,K1+1)=P(M,K1)
  160         CONTINUE
              P(M,2)=0
              GO TO 130
  150         DO 180 JJ=2,WWW
                II=WWW-JJ+2
                P(M,II)=X(LL,II)+P(M,II)+SS(M,II)
                IF(P(M,II).LT.2)GO TO 180
                P(M,II)=P(M,II)-2
                SS(M,II-1)=1
  180         CONTINUE
              IF(SS(M,1).EQ.0)GO TO 121
  764         DO 528 K=2,WWW
                K1=WWW-K+2
                P(M,K1+1)=P(M,K1)
  528         CONTINUE
              P(M,2)=1
              GO TO 121
  130       CONTINUE
C
C***********END OF LOOP 130*************
            DO 190 II=2,WW
  190         P(M,II)=P(M,II+1)
            IF(H(M,1).EQ.X(LL,1))GO TO 240
            P(M,1)=1
            GO TO 250
  240       P(M,1)=0
C*********THE BEGINING OF THE TWO'S COMPLEMENT OF P*********
C
  250       IF(P(M,1).EQ.0)GO TO 600
            DO 610 II=2,WWW
              IF(P(M,II).EQ.0)GO TO 620
              P(M,II)=0
              GO TO 610
  620         P(M,II)=1
  610       CONTINUE
            DO 602 II=1,WWW-1
              PP(M,II)=0
              SS(M,II)=0
  602       CONTINUE
```

```
          PP(M,WWW)=1
          SS(M,WWW)=0
          DO 603 II=2,WWW
            JJ=WWW-II+2
            P(M,JJ)=P(M,JJ)+PP(M,JJ)+SS(M,JJ)
            IF(P(M,JJ).LT.2)GO TO 603
            P(M,JJ)=P(M,JJ)-2
            SS(M,JJ-1)=1
603       CONTINUE
600       DO 201 II=1,WWW
            JJ=WWW-II+1
            P(M,JJ+1)=P(M,JJ)
201       CONTINUE
          P(M,1)=0
C
C***********END OF THE TWO'S COMPLEMENT OF P******
          DO 209 II=1,WW1
209         SS(M,II)=0
          DO 200 JJ=2,WW1
            II=WW1-JJ+1
            YY(J,II)=YY(J,II)+P(M,II)+SS(M,II)
            IF(YY(J,II).LT.2)GO TO 200
            YY(J,II)=YY(J,II)-2
            SS(M,II-1)=1
200       CONTINUE
          IF(SS(M,1).EQ.1)GO TO 781
          IF(SS(M,2).EQ.1)GO TO 781
          GO TO 184
781       DO 678 II=1,WWW
            JJJ=WWW-II+1
            YY(J,JJJ+1)=YY(J,JJJ)
678       CONTINUE
          YY(J,1)=0
184       IF(M.EQ.2)GO TO 798
          IF(LL.EQ.0)GO TO 798
          IF(LL.GT.J)GO TO 929
          GO TO 110
798       DO 183 II=1,WWW
183         YC(J,II)=YY(J,II+1)
          IF(YC(J,1).EQ.0)GO TO 800
          DO 810 II=2,WWW
            IF(YC(J,II).EQ.0)GO TO 820
            YC(J,II)=0
            GO TO 810
820         YC(J,II)=1
810       CONTINUE
          DO 819 II=1,WWW-1
            PP(J,II)=0
            SS(J,II)=0
819       CONTINUE
```

```
          PP(J,WWW)=1
          SS(J,WWW)=0
          DO 829 II=2,WWW
            JJ=WWW-II+2
            YC(J,JJ)=YC(J,JJ)+PP(J,JJ)+SS(J,JJ)
            IF(YC(J,JJ).LT.2)GO TO 829
            YC(J,JJ)=YC(J,JJ)-2
            SS(J,JJ-1)=1
  829     CONTINUE
  800     CONTINUE
          IF(RF.GT.0)GO TO 588
          WRITE(2,923)J,(YC(J,JJ),JJ=1,WWW)
          IF(J.EQ.(S-1))GO TO 654
          GO TO 932
  654     CALL CLOSE(2,IER)
          IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
          CALL CLOSE(1,IER)
          IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C*******END OF THE SECOND ORDER COMPONENT CONVOLUTION********
          GO TO 932
  588     WRITE(3,923)J,(YC(J,JJ),JJ=1,WWW)
  932     IF(J.NE.S-1)GO TO 110
          DO 934 JJ=1,CW
            H(0,JJ)=H(RF+3,JJ)
            H(1,JJ)=H(RF+4,JJ)
            H(2,JJ)=H(RF+5,JJ)
  934     CONTINUE
          RF=RF+3
          GO TO 412
  110     CONTINUE
  921   CONTINUE
        CALL CLOSE(3,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
        CALL CLOSE(2,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C       END CONVOLOTION OF CASCADE FORM
C
C*************************************************************
  915   FORMAT(2X,I5)
  916   FORMAT(1X,I5)
  910   FORMAT(4X,"I",5X,"UNQUANTIZED OUTPUT")
  911   FORMAT(4X,"-",5X,"------------------")
  923   FORMAT(1X,I4,3X,140(I1))
  929   STOP
        END
```

USER'S MANUAL PROGRAM POUT

FILE: POUT

DIRECTORY: DP4:OWEN

LANGUAGE: FORTRAN 5

DATE: September 1983

AUTHOR: Harun Inanli

SUBJECT: Calculating the Parallel Form Digital Filter Output Response

FUNCTION: This program computes the parallel form digital filter output response. Each second-order section coefficients and input signal values are taken from two different files in binary. Then, for each second-order section, they are multiplied and added based on convolution. The addition is carried out in two's complement. The input to all second-order sections is the same. The addition of all second-order sections will be the required output response for the parallel form. This response will be stored in binary.

PROGRAM USE: The program is loaded by the following command:

RLDR POUT @FLIB@

SUBROUTINE REQUIRED: None

FLOWGRAPH:

Type	Figure
1. Two's Complement of Binary Number	26
2. Two's Complement Addition	28
3. Binary Multiplication	29
4. Shift-left and Shift-right Operator	30
5. FIR Parallel Form Structure	33

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
POUT
BINARY COEFFICIENT FILE NAME:  TC
FIRST SECOND ORDER FILTER OUTPUT:  TO
BINARY INPUT FILE NAME:  TI
BINARY INPUT FILE NAME:  TI
NEXT SECOND ORDER OUTPUT FILE:  TO1
FIRST SECOND ORDER FILTER OUTPUT:  TO
ENTER THE FILE NAME FOR FIRST SECOND ORDER:  TO2
NEXT SECOND ORDER OUTPUT FILE:  TO1
FIRST SECOND ORDER OUTPUT FILE:  TO2
ENTER PARALLEL OUTPUT FILE STRUCTURE:  PTO
```

The content of the file TC and TI in Appendix B and the file TO in Program COUT are explained. The file TO1 and TO2 have the similar type of data as the file TO. The file PTO contains the output response of the parallel form structure in binary.

PTO

0	000000001111011101101000000000000
1	000000100101100100000011100100000
2	000000110101000001101011100100000
3	000000110101000001101011100100000
4	000000110101000001101011100100000
5	000000100101100100000011100100000
6	000000001111011101101000000000000
7	000000000000000000000000000000000
8	000000000000000000000000000000000
9	000000000000000000000000000000000

```
C*******************************************************************
C
C       PROGRAM :         POUT
C       AUTHOR   :        HARUN INANLI
C       DATE     :        SEPTEMBER 83
C       LANGUAGE:         FORTRAN 5
C
C       FUNCTION:         THIS PROGRAM IS USED TO FIND THE FILTER
C                         OUTPUT BASED ON CONVOLUTION BY USING THE PARALEL
C                         FILTER STRUCTURE  THE NEGATIVE NUMBER IS
C                         REPERESENTED IN TWO'S COMPLEMENT. THEN SUMMATION
C                         IS CARRIED OUT IN THIS NUMBER SYSTEM, TOO.
C                         THE OUTPUT VALUES IS STORED IN THE FILE.
C                         EACH COMPONENT IS THE SECOND DEGREE FILTER
C
C*********************************************************************

        INTEGER OUTFILE(7),OUTF(7),OUTD(7),OUTA(7),OUTFM(7)
        INTEGER X(0:20,140),H(0:20,140),PP(0:20,140),YC(0:20,140)
        INTEGER P(0:20,140),SS(0:20,140),YY(0:20,140)
        INTEGER IW,NC,CW,S,F,RF,RR,JB,JA,QQ
C********BINERY FILTER COEFFICIENTS ARE READ BY MEANS**********
C               OF CHANNEL (1)
C
        ACCEPT"BINERY COEFFICIEN FILE NAME : "
        READ(11,50)OUTFILE(1)
   50   FORMAT(S15)
        CALL OPEN(1,OUTFILE,1,IER)
        READ(1,60)CW
   60   FORMAT(20X,I5)
        READ(1,60)NC
        DO 70 I=0,(NC-1)
   70     READ(1,80)(H(I,K),K=1,CW)
   80   FORMAT(12X,140(I1))
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C***********COEFFICIENT**********************
   10   FORMAT(S15)
   30   FORMAT(20X,I5)
C*********FIRST SECOND ORDER FILTER OUTPUT IS*********
C                STORED IN THE FILE BY MEANS OF
C                CHANNEL (2)
C
        ACCEPT "FIRST SECOND ORDER FILTER OUTPUT : "
        READ(11,905)OUTF(1)
  905   FORMAT(S15)
        CALL DFILW(OUTF,IER)
        IF(IER.EQ.13)GO TO 906
        IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
```

```
906     CALL CFILW(OUTF,2,IER)
        IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
        CALL OPEN(2,OUTF,3,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
        RF=0
C*******THE INPUT TO THE FILTER IS READ FROM********
C                THE FILE BY MEANS OF CHANNEL(1)
C
412     ACCEPT"BINERY INPUT FILE NAME : "
        READ(11,10)OUTFILE(1)
        CALL OPEN(1,OUTFILE,1,IER)
        IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
        IF(RF.EQ.0)GO TO 578
        REWIND 1
578     READ(1,30)S
        READ(1,30)IW
        WW=2*IW
        WWW=2*IW+1
        IWW=IW+1
        WW1=2*IW+2
        CWW=CW+1
        DO 400 I=0,(S-1)
          DO 410 K=IWW,WWW
            X(I,K)=0
            XA(I,K)=0
410       CONTINUE
          DO 401 M=0,(NC-1)
            IF(M.GT.I)GO TO 400
            DO 402 K=1,WWW+2
              SS(M,K)=0
402         CONTINUE
401       CONTINUE
400     CONTINUE
        DO 430 M=0,(NC-1)
          DO 440 K=CWW,WWW
440         H(M,K)=0
430     CONTINUE
40      FORMAT(12X,140(I1))
C***************************************************************
C
C       THE BEGINING OF CONVOLUTION FOR EACH SECOND
C               ORDER FILTER
C
        J=0
        IF(RF.EQ.0)GO TO 513
```

```
C*******NEXT SECOND ORDER FILTER OUTPUT IS STORED IN THE******
C                   FILE BY MEANS OF CHANNEL(3)
C
        ACCEPT "NEXT SECOND ORDER OUTPUT FILE : "
        READ(11,10)OUTD(1)
        CALL DFILW(OUTD,IER)
        IF(IER.EQ.13)GO TO 584
        IF(IER.NE.1)TYPE "DELETE FILE ERROR",IER
  584   CALL CFILW(OUTD,2,IER)
        IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
        CALL OPEN(3,OUTD,3,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
  513   RR=0
        JB=0
  433   JB=J
        DO 435 J=JB,(JB+9)
          DO 436 K5=1,WW1
            YY(J,K5)=0
            YC(J,K5)=0
  436     CONTINUE
  435   CONTINUE
        IF(JB.EQ.297)GO TO 467
        IF(JB.EQ.198)GO TO 467
        IF(JB.EQ.99)GO TO 467
        IF(RR.EQ.400)GO TO 458
        IF(RR.EQ.300)GO TO 458
        IF(RR.EQ.200)GO TO 458
        IF(RR.EQ.100)GO TO 458
C**********THE LOOP 20 IS USED TO READ INPUT***********
C
  467   DO 20 JA=RR,(RR+9)
  20      READ(1,40,END=41,ERR=929)(X(JA,K),K=1,IW)
  41    CONTINUE
C
C**********END OF LOOP 20**********************
  458   RR=JA
        DO 921 J=JB,(JB+9)
          IF(RF.GT.(NC-1))GO TO 196
          IF(J.GT.(S-1))GO TO 932
          DO 110 M=0,2
            LL=J-M
            IF(LL.LT.0)GO TO 921
            IF(J.GE.(JB+9))GO TO 433
            DO 960 II=1,WWW+2
              P(M,II)=0
              SS(M,II)=0
  960       CONTINUE
```

```
C*********THE LOOP 130 IS USED FOR BINERY MILTIPLICATION********
C
            DO 130 R=2,CW
              KK=CW-R+2
              IF(H(M,KK).EQ.1)GO TO 150
  121         DO 160 K=2,WWW
                K1=WWW-K+2
                P(M,K1+1)=P(M,K1)
  160         CONTINUE
              P(M,2)=0
              GO TO 130
  150         DO 180 JJ=2,WWW
                II=WWW-JJ+2
                P(M,II)=X(LL,II)+P(M,II)+SS(M,II)
                IF(P(M,II).LT.2)GO TO 180
                P(M,II)=P(M,II)-2
                SS(M,II-1)=1
  180         CONTINUE
              IF(SS(M,1).EQ.0)GO TO 121
  764         DO 528 K=2,WWW
                K1=WWW-K+2
                P(M,K1+1)=P(M,K1)
  528         CONTINUE
              P(M,2)=1
              GO TO 121
  130       CONTINUE
C
C**********END OF LOOP 130*************
            DO 190 II=2,WW
  190         P(M,II)=P(M,II+1)
            IF(H(M,1).EQ.X(LL,1))GO TO 240
            P(M,1)=1
            GO TO 250
  240       P(M,1)=0
C*********THE BEGINING OF THE TWO'S COMPLEMENT OF P*********
C
  250       IF(P(M,1).EQ.0)GO TO 600
            DO 610 II=2,WWW
              IF(P(M,II).EQ.0)GO TO 620
              P(M,II)=0
              GO TO 610
  620         P(M,II)=1
  610       CONTINUE
            DO 602 II=1,WWW-1
              PP(M,II)=0
              SS(M,II)=0
  602       CONTINUE
            PP(M,WWW)=1
            SS(M,WWW)=0
            DO 603 II=2,WWW
              JJ=WWW-II+2
              P(M,JJ)=P(M,JJ)+PP(M,JJ)+SS(M,JJ)
              IF(P(M,JJ).LT.2)GO TO 603
              P(M,JJ)=P(M,JJ)-2
              SS(M,JJ-1)=1
  603       CONTINUE
```

```
600       DO 201 II=1,WWW
            JJ=WWW-II+1
            P(M,JJ+1)=P(M,JJ)
201       CONTINUE
          P(M,1)=0
C
C***********END OF THE TWO'S COMPLEMENT OF P******
C
C*******THIS PART IS USED FOR BINERY ADDITION************
C
          DO 209 II=1,WW1
209         SS(M,II)=0
          DO 200 JJ=2,WW1
            II=WW1-JJ+1
            YY(J,II)=YY(J,II)+P(M,II)+SS(M,II)
            IF(YY(J,II).LT.2)GO TO 200
            YY(J,II)=YY(J,II)-2
            SS(M,II-1)=1
200       CONTINUE
          IF(SS(M,1).EQ.1)GO TO 781
          IF(SS(M,2).EQ.1)GO TO 781
          GO TO 184
781       DO 678 II=1,WWW
            JJJ=WWW-II+1
            YY(J,JJJ+1)=YY(J,JJJ)
678       CONTINUE
          YY(J,1)=0
C
C**********ADDITION****************************
184       IF(M.EQ.2)GO TO 798
          IF(LL.EQ.0)GO TO 798
          GO TO 110
798       DO 183 II=1,WWW
183         YC(J,II)=YY(J,II+1)
          IF(YC(J,1).EQ.0)GO TO 800
          DO 810 II=2,WWW
            IF(YC(J,II).EQ.0)GO TO 820
            YC(J,II)=0
            GO TO 810
820         YC(J,II)=1
810       CONTINUE
          DO 819 II=1,WWW-1
            PP(J,II)=0
            SS(J,II)=0
819       CONTINUE
          PP(J,WWW)=1
          SS(J,WWW)=0
          DO 829 II=2,WWW
            JJ=WWW-II+2
            YC(J,JJ)=YC(J,JJ)+PP(J,JJ)+SS(J,JJ)
            IF(YC(J,JJ).LT.2)GO TO 829
            YC(J,JJ)=YC(J,JJ)-2
            SS(J,JJ-1)=1
829       CONTINUE
800       CONTINUE
```

```
            IF(RF.GT.0)GO TO 588
            WRITE(2,923)J,(YC(J,JJ),JJ=1,WWW)
            IF(J.EQ.(S-1))GO TO 654
            GO TO 932
  654       CALL CLOSE(2,IER)
            IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C*********WRITTEN IS COMPLETED FOR FIRST SECOND ORDER FILTER********
            CALL CLOSE(1,IER)
            IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C********READ IS COMPLETED FOR INPUT TO THE FILTER*******
            GO TO 932
  588       WRITE(3,923)J,(YC(J,JJ),JJ=1,WWW)
            IF(J.EQ.(S-1))GO TO 359
  932       IF(J.NE.S-1)GO TO 110
            DO 934 JJ=1,CW
              H(0,JJ)=H(RF+3,JJ)
              H(1,JJ)=H(RF+4,JJ)
              H(2,JJ)=H(RF+5,JJ)
  934       CONTINUE
            RF=RF+3
            IF(RF.GT.3)GO TO 196
            GO TO 412
  110     CONTINUE
  921   CONTINUE
  359   CALL CLOSE(3,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C**********WRITTEN IS COMPLETED FOR SECOND SECOND ORDER FILTER*****
C
C       END OF CONVOLOTION OF EACH SECOND ORDER FILTER
C
C******************************************************************
  915   FORMAT(2X,I5)
  916   FORMAT(1X,I5)
  910   FORMAT(4X,"I",5X,"UNQUANTIZED OUTPUT")
  911   FORMAT(4X,"-",5X,"------------------")
  923   FORMAT(1X,I4,3X,140(I1))
C*******FIRST SECOND ORDER FILTER OUTPUT IS READ******
C               FROM THE FILE BY MEANS OF
C               CHANNEL(2)
C
  196   ACCEPT"FIRST SECOND ORDER FILTER OUTPUT : "
        READ(11,905)OUTFILE(1)
        CALL OPEN(2,OUTFILE,1,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
        REWIND 2
        ACCEPT"ENTER THE FILE NAME FOR FIRST SECOND ORDER : "
        READ(11,10)OUTFM(1)
        CALL DFILW(OUTFM,IER)
```

```
      IF(IER.EQ.13)GO TO 386
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
386   CALL CFILW(OUTFM,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(6,OUTFM,3,IER)
      IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
      QQ=0
      J=0
      JA=0
312   RR=0
      JB=0
      IF(JB.EQ.0)GO TO 354
221   JB=J+1
354   RR=JA
      IF(QQ.NE.0)GO TO 316
C*********THE LOOP 192 IS USED TO READ THE FIRST SECOND********
C                ORDER OUTPUT
C
      DO 192 JA=RR,(RR+9)
        DO 213 JJ=1,WWW
213       YY(JA,JJ)=0
        READ(2,923,END=193,ERR=929)J,(YY(JA,K5),K5=1,WWW)
192   CONTINUE
193   CONTINUE
C
C***********END OF LOOP 192***************************
      DO 214 JL=JB,(JB+9)
        DO 215 JJ=1,WWW
          YC(JL,JJ)=0
          SS(JL,JJ)=0
215     CONTINUE
214   CONTINUE
      GO TO 313
316   IF(J.GE.9)GO TO 364
C*********THE OUTPUT OF THE NEXT SECOND ORDER FILTER************
C                IS READ FROM THE FILE BY MEANS OF
C                CHANNEL(3)
C
      ACCEPT"NEXT SECOND ORDER OUTPUT FILE : "
      READ(11,10)OUTD(1)
      CALL OPEN(3,OUTD,1,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      REWIND 3
C***********THE OUTPUT OF THE FIRST SECOND ORDER FILTER**********
C                IS READ FROM THE FILE BY MEANS OF
C                CHANNEL(6)
      ACCEPT"FIRST SECOND ORDER OUTPUT FILE : "
      READ(11,905)OUTFM(1)
      CALL OPEN(6,OUTFM,1,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      REWIND 6
```

```
C*********THE OUTPUT OF THE PARALLEL STRUCTURE FILTER IS***********
C                   WRITTEN TO THE FILE BY MEANS OF
C                   CHANNEL(5)
        ACCEPT"ENTER PARALEL OUTPUT FILE STRUCTURE : "
        READ(11,905)OUTA(1)
        CALL DFILW(OUTA,IER)
        IF(IER.EQ.13)GO TO 365
        IF(IER.NE.1)TYPE "DELETE FILE ERROR",IER
  365   CALL CFILW(OUTA,2,IER)
        IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
        CALL OPEN(5,OUTA,3,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
C**********THE LOOP 323 IS USED TO READ THE FIRST*********
C                   AND SECOND ORDER OUTPUT FILTER
C
  364   DO 323 JA=RR,(RR+9)
          DO 366 JJ=1,WWW
  366       YY(JA,JJ)=0
          IF(JA.GT.(S-1))GO TO 929
          READ(3,923,END=324,ERR=929)J,(YY(JA,K9),K9=1,WWW)
          READ(6,923,END=324,ERR=929)J,(YC(JA,KK5),KK5=1,WWW)
  323   CONTINUE
  324   CONTINUE
C
C*********END OF LOOP 323************
        DO 314 J=JB,(JB+9)
          DO 315 JJ=1,WWW
  315       SS(J,JJ)=0
  314   CONTINUE
  313   DO 194 J=JB,(JB+9)
          DO 195 K=2,WWW
            JJ=WWW-K+1
            YC(J,JJ)=YC(J,JJ)+YY(J,JJ)+SS(J,JJ)
            IF(YC(J,JJ).LT.2)GO TO 195
            YC(J,JJ)=YC(J,JJ)-2
            SS(J,JJ-1)=1
  195     CONTINUE
          IF(SS(J,1).EQ.1)GO TO 216
          IF(SS(J,2).EQ.1)GO TO 216
          GO TO 217
  216     DO 218 JJ=1,WWW
            II=WWW-JJ+1
            YC(J,II+1)=YC(J,II)
  218     CONTINUE
  217     IF(QQ.EQ.0)GO TO 369
          WRITE(5,923)J,(YC(J,JJ),JJ=1,WWW)
          GO TO 388
  369     WRITE(6,923)J,(YC(J,JJ),JJ=1,WWW)
  388     IF(J.GE.(S-1))GO TO 311
          IF(J.GE.(JB+9))GO TO 221
  194   CONTINUE
```

```
311     QQ=QQ+1
        J=-1
        JB=0
        JA=0
        CALL CLOSE(6,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C          WRITTEN OF THE FIRST SECOND ORDER FILTER
C*******************IS COMPLETED***************
        IF(QQ.GE.2)GO TO 373
        GO TO 312
373     CALL CLOSE(5,IER)
        IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
C
C       WRITTEN OF THE PARALLEL FILTER OUTPUT
C**************IS COMPLETED**********************
929     STOP
        END
```

USER'S MANUAL PROGRAM NES

FILE:	TNES
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Calculating the Nested Filter Output Response.

FUNCTION: This program is used to calculate the nested filter output response based on the equation below:

Y(N) = H(O)(X(N) + H(1)(X(N-1)

+ ... + H(M)X(N-M))...)

where N and M = number of input and coefficient, respectively; Y = output; X = input; and H = coefficient.

The filter coefficients and inputs are taken from two different files. The necessary addition is carried out in two's complement. Then, the output will be stored in binary.

PROGRAM USE: The program is loaded by the following command:

RLDR TNES @FLIB@

SUBROUTINE REQUIRED: None

FLOWGRAPH:

Type	Figure
1. Two's Complement of Binary Numbers	26
2. Two's Complement Addition	28
3. Binary Multiplication	29
4. Shift-left and Shift-right	30
5. FIR Nested Form Structure	34

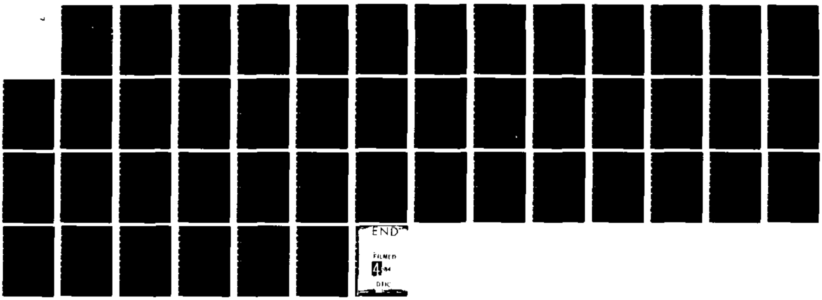
EXECUTION OF THE PROGRAM AND ITS RESULTS:

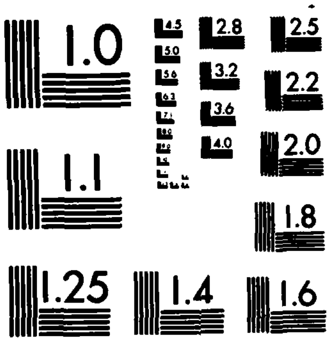
```
TNES
NESTED STRUCTURE BINARY COEFFICIENT FILE NAME:  NC
BINARY INPUT FILE NAME:  TI
UNQUANTIZE BINARY OUTPUT NAME FOR NS:  NO
```

The contents of the file TI in Appendix B is explained. The file NC which has very similar data to the file TC explained before, represents the nested filter coefficients in binary. The file NO, representing the Nested filter output response, has also the similar data explained in Program TO.

AD-A138 082 STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. H INANLI 3/3

UNCLASSIFIED DEC 83 AFIT/GE/EE/83D-32 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
C**************************************************************
C
C         PROGRAM :         NES
C         AUTHOR  :         HARUN INANLI
C         DATE    :         SEPTEMBER 83
C         LANGUAGE:         FORTRAN 5
C
C         FUNCTION:         THIS PROGRAM IS USED TO CALCULATE THE NESTED
C                           FILTER OUTPUT IN BINERY. THE INPUTS TO THIS
C                           PROGRAM ARE TAKEN FROM THE FILES. THEY CONTAIN
C                           NESTED STRUCTURE COEFFICIENTS AND INPUT VALUES
C                           IN BINERY THE OUTPUT OF THE NESTED STRUCTURE IS
C                           STORED IN THE FILE IN BINERY SUCH THAT WORD LENGT
C                           OF THE OUTPUT TWO TIMES BIGGER THEN THE WORD
C                           LENGTH OF THE INPUT.
C
C*****************************************************************
          INTEGER OUTFILE(7),OUTF(7),XX(20,140),Y(20,140),X1(20,140)
          INTEGER X(20,140),H(20,140),P(20,140),SS(20,140),PP(20,140)
          INTEGER IW,NC,CW,S,J,I,J1,R,K,II,KKK,F,Q,IA,IB,IC,WW1,CWW
C*******THIS PART IS USED TO READ THE NESTED STRUCTURE**************
C                   COEFFICIENT.
C
          ACCEPT"NESTED STRUCTURE BINERY COEFFICIEN FILE NAME : "
          READ(11,50)OUTFILE(1)
   50     FORMAT(S15)
          CALL OPEN(1,OUTFILE,1,IER)
          READ(1,60)NC
          READ(1,60)CW
   60     FORMAT(5X,I4)
          DO 200 IJ=0,(NC-1)
            DO 201 JJ=1,(2*CW+1)
  201         H(IJ,JJ)=0
  200     CONTINUE
          I=0
          DO 70 I=0,(NC-1)
   70       READ(1,80)(Q,(H(I,K),K=1,CW))
   80     FORMAT(1X,I4,10X,140(I1))
          CALL CLOSE(1,IER)
          IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C*********NESTED COEFFICIENT*********************
C
C*********CHANNEL (1) IS USED TO READ THE INPUT*******
C                   FROM THE FILE
C
          ACCEPT"BINERY INPUT FILE NAME : "
          READ(11,10)OUTFILE(1)
   10     FORMAT(S15)
          CALL OPEN(1,OUTFILE,1,IER)
          IF(IER.NE.1)TYPE"OPEN INPUT FILE ERROR : ",IER
          READ(1,30) S
   30     FORMAT(20X,I5)
          READ(1,30)IW
```

```
C*******CHANNEL(1) UNDER THE NAME OF OUTF IS USED TO WRITE*********
C                  THE OUTPUT VALUES
C
        ACCEPT"UNQUANTIZED BINERY OUTPUT NAME FOR NS : "
        READ(11,100)OUTF(1)
 100    FORMAT(S15)
        CALL DFILW(OUTF,IER)
        IF(IER.EQ.13)GO TO 101 ;
        IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
 101    CALL CFILW(OUTF,2,IER)
        IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
        CALL OPEN(2,OUTF,3,IER)
        IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
        WRITE(2,980)IW
        WRITE(2,981)S
 980    FORMAT(2X,I5)
 981    FORMAT(1X,I5)
        WW=2*IW
        WWW=2*IW+1
        IWW=IW+1
        WW1=2*IW+2
        CWW=CW+1
        IB=0
        IA=0
        IC=0
        R=0
C*******THE LOOP 400 IS USED TO FIND THE OUTPUT************
C                 FOR EACH SAMPLE
C
 400    I=R
        IF(I.EQ.360)GO TO 434
        IF(I.EQ.300)GO TO 434
        IF(I.EQ.240)GO TO 434
        IF(I.EQ.180)GO TO 434
        IF(I.EQ.120)GO TO 434
        IF(I.EQ.60)GO TO 434
        IF(IA.EQ.360)GO TO 433
        IF(IA.EQ.300)GO TO 433
        IF(IA.EQ.240)GO TO 433
        IF(IA.EQ.180)GO TO 433
        IF(IA.EQ.120)GO TO 433
        IF(IA.EQ.60)GO TO 433
C******THE LOOP 20 IS USED TO READ THE INPUT************
C                 10 AT A TIME
C
 434    DO 20 J=IA,(IA+9)
 20       READ(1,40,END=41)(X(J,KK),KK=1,IW)
 41     CONTINUE
C
C*******END OF LOOP 20***********************
```

```
C********THIS PART IS USED TO FIND THE Y(O)*************
C
  433     IF(IB.EQ.1)GO TO 412
          DO 356 JJ=(IW+1),WWW
  356       X(O,JJ)=0
          DO 413 JJ=1,WWW
            Y(O,JJ)=0
            SS(O,JJ)=0
  413     CONTINUE
          DO 414 N=2,CW
            KK=CW-N+2
            IF(H(O,KK).EQ.1)GO TO 415
  418       DO 416 JJ=2,WWW
              K1=WWW-K+2
              Y(O,K1+1)=Y(O,K1)
  416       CONTINUE
            Y(O,2)=0
            GO TO 414
  415       DO 417 JJ=2,WWW
              JJJ=WWW-JJ+2
              Y(O,JJJ)=Y(O,JJJ)+SS(O,JJJ)+X(O,JJJ)
              IF(Y(O,JJJ).LT.2)GO TO 417
              Y(O,JJJ)=Y(O,JJJ)-2
              SS(O,JJJ-1)=1
  417       CONTINUE
            IF(SS(O,1).EQ.O)GO TO 418
            DO 419 K=2,WWW
              K1=WWW-K+2
              Y(O,K1+1)=Y(O,K1)
  419       CONTINUE
            Y(O,2)=1
            GO TO 418
  414     CONTINUE
          WRITE(2,923)O,(Y(O,JJ),JJ=1,WWW)
          IB=1
C
C*******Y(O) IS WRITTEN INTO THE FILE**********************
  412     IA=J
C*********THE LOOP 401 IS USED TO FIND THE OUTPUT***********
C                   9 AT A TIME
C
          DO 401 R=I,(I+9)
            IF(R.EQ.S)GO TO 500
            IF(R.EQ.IA)GO TO 400
            DO 501 L=1,WW1
              XX(R,L)=0
  501       CONTINUE
            IF(R.GT.(NC-1))GO TO 310
            KKK=R
            F=0
            GO TO 312
  310       KKK=NC
            F=R-NC
```

```
310       DO 355 JJ=(IW+1),WWW
355         X(R,JJ)=0
          DO 778 JJ=1,WWW
778         X1(R,JJ)=X(R,JJ)
          IC=II
          IF(R.GE.(I+9))GO TO 400
C*********THE LOOP 110 IS USED TO FIND THE OUTPUT*******
C                 1 AT A TIME.
C
          DO 110 II=F,F+NC-1
            IF(KKK.GT.(NC-1))GO TO 444
            J1=KKK-II
            GO TO 449
444         J1=R-II
449         IF(J1.LE.0)GO TO 401
            IF(J1.GE.NC)GO TO 110
            DO 560 JJ=IWW,WWW
560           H(J1,JJ)=0
            DO 111 JJ=1,WWW
              SS(II,JJ)=0
              P(II,JJ)=0
111         CONTINUE
C**********THE LOOP 112 IS USED FOR BINERY MULTIPLICATION*********
C
            DO 112 N=2,WWW
              KK=WWW-N+2
              IF(H(J1,KK).EQ.1)GO TO 113
116           DO 114 K=2,WWW
                K1=WWW-K+2
                P(II,K1+1)=P(II,K1)
114           CONTINUE
              P(II,2)=0
              GO TO 112
113           DO 115 JJ=2,WWW
                JJJ=WWW-JJ+2
                P(II,JJJ)=P(II,JJJ)+X1(II,JJJ)+SS(II,JJJ)
                IF(P(II,JJJ).LT.2)GO TO 115
                P(II,JJJ)=P(II,JJJ)-2
                SS(II,JJJ-1)=1
115           CONTINUE
              IF(SS(II,1).EQ.0)GO TO 116
              DO 900 K=2,WWW
                K1=WWW-K+2
                P(II,K1+1)=P(II,K1)
900           CONTINUE
              P(II,2)=1
              GO TO 116
112         CONTINUE
C
C*********END OF LOOP 112****************************************
            DO 669 JJ=2,WWW
669           P(II,JJ)=P(II,JJ+1)
            IF(H(J1,1).EQ.X1(II,1))GO TO 118
            P(II,1)=1
            GO TO 119
118         P(II,1)=0
```

```
C************THE BEGINING OF THE TWO'S COMPLEMENT OF P******
C
  119       IF(P(II,1).EQ.0)GO TO 120
            DO 121 JJ=2,WWW
              IF(P(II,JJ).EQ.0)GO TO 122
              P(II,JJ)=0
              GO TO 121
  122         P(II,JJ)=1
  121       CONTINUE
            DO 130 JJ=1,WWW-1
              PP(II,JJ)=0
              SS(II,JJ)=0
  130       CONTINUE
            PP(II,WWW)=1
            SS(II,WWW)=0
            DO 131 JJ=2,WWW
              JJJ=WWW-JJ+2
              P(II,JJJ)=P(II,JJJ)+PP(II,JJJ)+SS(II,JJJ)
              IF(P(II,JJJ).LT.2)GO TO 131
              P(II,JJJ)=P(II,JJJ)-2
              SS(II,JJJ-1)=1
  131       CONTINUE
C
C************TWO'S COMPLEMENT OF P**************************
C
C************THE BEGINING OF THE TWO'S COMPLEMENT OF X********
C
  120       IF(X1(II+1,1).EQ.0)GO TO 123
            DO 124 JJ=2,WWW
              IF(X1(II+1,JJ).EQ.0)GO TO 126
              X1(II+1,JJ)=0
              GO TO 124
  126         X1(II+1,JJ)=1
  124       CONTINUE
            DO 135 JJ=1,WWW-1
              PP(II,JJ)=0
              SS(II,JJ)=0
  135       CONTINUE
            PP(II,WWW)=1
            SS(II,WWW)=0
            DO 136 JJ=2,WWW
              JJJ=WWW-JJ+2
              X1(II+1,JJJ)=X1(II+1,JJJ)+PP(II,JJJ)+SS(II,JJJ)
              IF(X1(II+1,JJJ).LT.2)GO TO 136
              X1(II+1,JJJ)=X1(II+1,JJJ)-2
              SS(II,JJJ-1)=1
  136       CONTINUE
C
C***********TWO'S COMPLEMENT OF X*****************
  123       DO 137 JJ=1,WWW
              JJJ=WWW-JJ+1
              X1(II+1,JJJ+1)=X1(II+1,JJJ)
  137       CONTINUE
            X1(II+1,1)=0
```

```
C************THE BEGINING OF THE TWO'S COMPLEMENT ADDITION*********
C
          DO 138 JJ=1,WW1
138         SS(II,JJ)=0
          DO 140  JJ=2,WW1
            JJJ=WW1-JJ+1
            XX(R,JJJ)=X1(II+1,JJJ)+P(II,JJJ)+SS(II,JJJ)
            IF(XX(R,JJJ).LT.2)GO TO 140
            XX(R,JJJ)=XX(R,JJJ)-2
            SS(II,JJJ-1)=1
140       CONTINUE
          IF(SS(II,1).EQ.1)GO TO 949
          IF(SS(II,2).EQ.1)GO TO 949
          DO 948 JJ=1,WWW
948         XX(R,JJ)=XX(R,JJ+1)
949       IF(XX(R,1).EQ.0)GO TO 678
          DO 148 JJ=2,WWW
            IF(XX(R,JJ).EQ.0)GO TO 149
            XX(R,JJ)=0
            GO TO 148
149         XX(R,JJ)=1
148       CONTINUE
          DO 150 JJ=1,WWW-1
            PP(R,JJ)=0
            SS(R,JJ)=0
150       CONTINUE
          PP(R,WWW)=1
          SS(R,WWW)=0
          DO 151 JJ=2,WWW
            JJJ=WWW-JJ+2
            XX(R,JJJ)=XX(R,JJJ)+PP(R,JJJ)+SS(R,JJJ)
            IF(XX(R,JJJ).LT.2) GO TO 151
            XX(R,JJJ)=XX(R,JJJ)-2
            SS(R,JJJ-1)=1
151       CONTINUE
C
C**********TWO'S COMPLEMENT ADDITION************************
678       DO 743 JJ=1,WWW
743         X1(II+1,JJ)=XX(R,JJ)
          DO 695 JJ=1,WWW
695         XX(R,JJ)=0
          IF(II.EQ.(R-1))GO TO 153
          GO TO 110
153       DO 610 JJ=1,WW1
            Y(R,JJ)=0
            SS(R,JJ)=0
610       CONTINUE
          DO 600 N=2,CW
            KK=CW-N+2
            IF(H(0,KK).EQ.1)GO TO 601
604         DO 602 K=2,WWW
              K1=WWW-K+2
              Y(R,K1+1)=Y(R,K1)
602         CONTINUE
```

```
          Y(R,2)=0
          GO TO 600
601       DO 603 JJ=2,WWW
            JJJ=WWW-JJ+2
            Y(R,JJJ)=Y(R,JJJ)+SS(R,JJJ)+X1(II+1,JJJ)
            IF(Y(R,JJJ).LT.2)GO TO 603
            Y(R,JJJ)=Y(R,JJJ)-2
            SS(R,JJJ-1)=1
603       CONTINUE
          IF(SS(R,1).EQ.0)GO TO 604
          DO 933 K=2,WWW
            K1=WWW-K+2
            Y(R,K1+1)=Y(R,K1)
933       CONTINUE
          Y(R,2)=1
          GO TO 604
600     CONTINUE
        DO 690 JJ=2,WWW
690       Y(R,JJ)=Y(R,JJ+1)
        IF(H(0,1).EQ.X1(II+1,1))GO TO 620
        Y(R,1)=1
        GO TO 621
620     Y(R,1)=0
621     WRITE(2,923)R,(Y(R,JJ),JJ=1,WWW)
        DO 888 B=F,(F+NC-1)
          DO 777 JJ=1,WWW
777         X1(B+1,JJ)=X(B+1,JJ)
888     CONTINUE
110   CONTINUE
C
C********END OF LOOP 110****************************
 401  CONTINUE
C
C**************END OF LOOP 401**************************
 40   FORMAT(12X,140(I1))
 923  FORMAT(1X,I4,3X,140(I1))
      CALL CLOSE (1,IER)
      IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
      CALL CLOSE(2,IER)
      IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
 500  CALL EXIT
      END
```

USER'S MANUAL PROGRAM CNES

FILE:	CNES
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Calculating the Cascade-Nested Filter Output Response.
FUNCTION:	This program computes the cascade-nested filter output response. Each second-order section is acting as an individual nested filter. The output of the first second-order section will be the input to the next section. The final second-order section output will be the output response to the cascade-nested structure. The necessary addition is carried out in two-s complement and the output will be stored in binary.
PROGRAM USE:	The program is loaded by the following command:
	RLDR CNES @FLIB@
SUBROUTINE REQUIRED:	None

FLOWGRAPH:

Type	Figure
1. Two's Complement of Binary Number	26
2. Two's Complement Addition	28
3. Binary Multiplication	29
4. Shift-left and Shift-right	30
5. FIR Cascade-Nested Form Structure	35

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
CNES
NESTED STRUCTURE BINARY COEFFICIENT FILE NAME:  NC
```

```
BINARY INPUT FILE NAME:  TI
UNQUANTIZE BINARY OUTPUT NAME FOR NS:  NO
ENTER THE NEXT SECOND ORDER SECTION:  NO
NEXT SECOND ORDER OUTPUT FILE:  CNO
```

The content of the file NC and the file NO in Program NEX and the TI in Appendix B are explained. The file CNO, representing the cascade-nested form output response, has the similar data to the file CTO explained in Program COUT.

```
C***********************************************************************
C        PROGRAM  :        CNES
C        AUTHOR   :        HARUN INANLI
C        DATE     :        SEPTEMBER 83
C        LANGUAGE :        FORTRAN 5
C
C        FUNCTION:         THIS PROGRAM IS USED TO CALCULATE THE FILTE
C                          OUTPUT BASED ON CASCADE-NESTED STRUCTURE
C                          THAT IS, EACH SECOND ORDER COMPONETS OF THE
C                          CCASCADE FILTER ARE IN NESTED FORM THE NEGA
C                          NUMBER IS REPERESENTED IN TWO'S COMPLEMENT
C                          SUMMATION IS CARRIED OUT IN THIS NUMBER SYS
C
C***********************************************************************
      INTEGER OUTFILE(7),OUTF(7),XX(20,140),Y(20,140),X1(20,140)
      INTEGER X(20,140),H(20,140),P(20,140),SS(20,140),PP(20,140)
      INTEGER IW,NC,CW,S,J,I,J1,R,K,II,KKK,F,RF,Q,QQ,OUTD(7),CWW
      ACCEPT"NESTED STRUCTURE BINERY COEFFICIEN FILE NAME : "
      READ(11,50)OUTFILE(1)
 50   FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      READ(1,60)NC
      READ(1,60)CW
 60   FORMAT(5X,I4)
      DO 200 IJ=0,(NC-1)
        DO 201 JJ=1,(2*CW+1)
 201      H(IJ,JJ)=0
 200  CONTINUE
C********BINARY NESTED FILTER COEFFICIENTS ARE READ BY*********
C                MEANS OF CHANNEL(1)
C
      DO 70 I=0,(NC-1)
 70     READ(1,80)(Q,(H(I,K),K=1,CW))
 80   FORMAT(1X,I4,10X,140(I1))
      CALL CLOSE(1,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C***********NESTED FILTER COEFFICIENT**********************
C
C**********THE INPUT TO THE FILTER IS READ FROM***********
C                THE FILE BY MEANS OF CHANNEL(1)
C
      ACCEPT"BINERY INPUT FILE NAME : "
      READ(11,10)OUTFILE(1)
 10   FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN INPUT FILE ERROR : ",IER
      READ(1,30) S
 30   FORMAT(20X,I5)
      READ(1,30)IW
C************FIRST SECOND ORDER FILTER OUTPUT IS **************
C                STORED IN THE FILE BY MEANS OF
C                CHANNEL(2)
C
      ACCEPT"UNQUANTIZED BINERY OUTPUT NAME FOR NS : "
      READ(11,100)OUTF(1)
 100  FORMAT(S15)
```

```
      CALL DFILW(OUTF,IER)
      IF(IER.EQ.13)GO TO 101
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
101   CALL CFILW(OUTF,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      UW=2*IW
      WWW=2*IW+1
      IWW=IW+1
      WW1=2*IW+2
      CWW=CW+1
C*****************************************************************
C
C        THE BEGINING OF THE CALCULATION OF THE OU   JT
C                 FOR CASCADE-NESTED STRUCTURE
C
      RF=0
      QQ=0
      IB=0
      IA=0
      IC=0
      R=0
      IF(RF.EQ.0)GO TO 513
      IF(RF.GT.(NC-1))GO TO 500
C********FIRST SECOND ORDER FILTER OUTPUT, WHICH*******
C                 IS INPUT TO THE NEXT SECOND ORDER
C                 FILTER, IS READ BY MEANS OF
C                 CHANNEL(2)
C
      ACCEPT"ENTER THE NEXT SECOND ORDER SECTION :  "
      READ(11,100)OUTF(1)
      CALL OPEN(2,OUTF,1,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      REWIND 2
C**********THE NEXT SECOND ORDER OUTPUT IS STORED*******
C                 IN THE FILE BY MEANS OF CHANNEL(3)
C
      ACCEPT"NEXT SECOND ORDER OUTPUT FILE :  "
      READ(11,100)OUTD(1)
      CALL DFILW(OUTD,IER)
      IF(IER.EQ.13)GO TO 584
      IF(IER.NE.1)TYPE "DELETE FILE ERROR",IER
584   CALL CFILW(OUTD,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(3,OUTD,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      WRITE(3,915)IW
      WRITE(3,916)S
513   CONTINUE
C*********THIS PART IS USED TO FIND THE OUTPUT OF***********
C                 EACH SECOND ORDER FILTER
C
500   I=R
      IF(RF.NE.0)GO TO 454
      IF(I.EQ.360)GO TO 434
      IF(I.EQ.300)GO TO 434
```

```
      IF(I.EQ.240)GO TO 434
      IF(I.EQ.180)GO TO 434
      IF(I.EQ.120)GO TO 434
      IF(I.EQ.60)GO TO 434
      IF(IA.EQ.360)GO TO 433
      IF(IA.EQ.300)GO TO 433
      IF(IA.EQ.240)GO TO 433
      IF(IA.EQ.180)GO TO 433
      IF(IA.EQ.120)GO TO 433
      IF(IA.EQ.60)GO TO 433
C*********THE LOOP 20 IS USED TO READ INPUT *************
C
 434  DO 20 J=IA,(IA+9)
 20     READ(1,40,END=41)(X(J,KK),KK=1,IW)
 41   CONTINUE
C
C*********END OF LOOP 20**********************
      GO TO 15
 433  IF(I.EQ.360)GO TO 16
      IF(I.EQ.300)GO TO 16
      IF(I.EQ.240)GO TO 16
      IF(I.EQ.180)GO TO 16
      IF(I.EQ.120)GO TO 16
      IF(I.EQ.60)GO TO 16
      IF(IA.EQ.360)GO TO 17
      IF(IA.EQ.300)GO TO 17
      IF(IA.EQ.240)GO TO 17
      IF(IA.EQ.180)GO TO 17
      IF(IA.EQ.120)GO TO 17
      IF(IA.EQ.60)GO TO 17
C********THE LOOP 452 IS USED TO READ THE INPUT OF**********
C                  NEXT SECOND ORDER FILTER
C
 15   DO 452 J=IA,(IA+9)
 452    READ(2,923,END=453)Q,(X(J,JJ),JJ=1,WWW)
 453  CONTINUE
C
C*********END OF LOOP 452************************
 17   CONTINUE
C*********THIS PART OF THE PROGRAM IS USED TO**********
C                  FIND THE Y(0)
C
 16   IF(IB.EQ.1)GO TO 412
      IB=1
      DO 356 JJ=IWW,WWW
 356    X(0,JJ)=0
      DO 413 JJ=1,WWW
        Y(0,JJ)=0
        SS(0,JJ)=0
 413  CONTINUE
      DO 414 N=2,CW
        KK=CW-N+2
        IF(H(0,KK).EQ.1)GO TO 415
 418    DO 416 K=2,WWW
          K1=WWW-K+2
          Y(0,K1+1)=Y(0,K1)
 416    CONTINUE
        Y(0,2)=0
        GO TO 414
```

```
      DO 417 JJ=2,WWW
        JJJ=WWW-JJ+2
        Y(0,JJJ)=Y(0,JJJ)+SS(0,JJJ)+X(0,JJ)
        IF(Y(0,JJJ).LT.2)GO TO 417
        Y(0,JJJ)=Y(0,JJJ)-2
        SS(0,JJJ-1)=1
417   CONTINUE
      IF(SS(0,1).EQ.0)GO TO 418
      DO 419 K=2,WWW
        K1=WWW-K+2
        Y(0,K1+1)=Y(0,K1)
419   CONTINUE
      Y(0,2)=1
      GO TO 418
414   CONTINUE
      DO 333 JJ=1,WWW
333     Y(0,JJ)=Y(0,JJ+1)
      IF(RF.NE.0)GO TO 455
      WRITE(2,923)QQ,(Y(0,JJ),JJ=1,WWW)
      GO TO 412
455   WRITE(3,923)QQ,(Y(0,JJ),JJ=1,WWW)
C
C*****COMPLITION OF THE Y(0)*****************************
412   IA=J
      DO 401 R=I,(I+9)
        IF(RF.EQ.6)GO TO 500
        IF(R.EQ.S)GO TO 486
        DO 501 L=1,WW1
501       XX(R,L)=0
        IF(R.GT.2)GO TO 310
        KKK=R
        F=0
        GO TO 312
310     KKK=2
        F=R-2
312     DO 355 JJ=IWW,WWW
355       X(R,JJ)=0
        DO 778 JJ=1,WWW
778       X1(R,JJ)=X(R,JJ)
        IF(R.GE.(I+9))GO TO 400
C*********THE LOOP 110 IS USED TO CALCULATE THE OUTPUT*********
C               OF EACH SECOND ORDER FILTER ONE BY ONE
C
        DO 110 II=F,(F+2)
          IF(KKK.GE.2)GO TO 444
          J1=KKK-II
          GO TO 449
444       J1=R-II
449       IF(J1.LE.0)GO TO 401
          DO 560 JJ=IWW,WWW
560         H(J1,JJ)=0
          DO 111 JJ=1,WWW
            SS(II,JJ)=0
            P(II,JJ)=0
111       CONTINUE
C*******THE LOOP 112 IS USED FOR BINERY MILTIPLICATION********
```

```
      DO 112 N=2,WWW
        KK=WWW-N+2
        IF(H(J1,KK).EQ.1)GO TO 113
116     DO 114 K=2,WWW
          K1=WWW-K+2
          P(II,K1+1)=P(II,K1)
114     CONTINUE
        P(II,2)=0
        GO TO 112
113     DO 115 JJ=2,WWW
          JJJ=WWW-JJ+2
          P(II,JJJ)=P(II,JJJ)+X1(II,JJJ)+SS(II,JJJ)
          IF(P(II,JJJ).LT.2)GO TO 115
          P(II,JJJ)=P(II,JJJ)-2
          SS(II,JJJ-1)=1
115     CONTINUE
        IF(SS(II,1).EQ.0)GO TO 116
        DO 200 K=2,WWW
          K1=WWW-K+2
          P(II,K1+1)=P(II,K1)
200     CONTINUE
        P(II,2)=1
        GO TO 116
112   CONTINUE
C***********END OF LOOP 112*********************
      DO 669 JJ=2,WWW
669     P(II,JJ)=P(II,JJ+1)
      IF(H(J1,1).EQ.X1(II,1))GO TO 118
      P(II,1)=1
      GO TO 119
118   P(II,1)=0
C************THE BEGINING OF THE TWO'S COMPLEMENT OF P**********

119   IF(P(II,1).EQ.0)GO TO 120
      DO 121 JJ=2,WWW
        IF(P(II,JJ).EQ.0)GO TO 122
        P(II,JJ)=0
        GO TO 121
122     P(II,JJ)=1
121   CONTINUE
      DO 130 JJ=1,WWW-1
        PP(II,JJ)=0
        SS(II,JJ)=0
130   CONTINUE
      PP(II,WWW)=1
      SS(II,WWW)=0
      DO 131 JJ=2,WWW
        JJJ=WWW-JJ+2
        P(II,JJJ)=P(II,JJJ)+PP(II,JJJ)+SS(II,JJJ)
        IF(P(II,JJJ).LT.2)GO TO 131
        P(II,JJJ)=P(II,JJJ)-2
        SS(II,JJJ-1)=1
131   CONTINUE

C*********END OF TWO'S COMPLEMENT OF P*************
```

```
C********THE BEGINING OF TWO'S COMPLEMENT OF X1(II+1)********
            IF(X1(II+1,1).EQ.0)GO TO 123
            DO 124 JJ=2,WWW
              IF(X1(II+1,JJ).EQ.0)GO TO 126
              X1(II+1,JJ)=0
              GO TO 124
126           X1(II+1,JJ)=1
124         CONTINUE
            DO 135 JJ=1,WWW-1
              PP(I1,JJ)=0
              SS(II,JJ)=0
135         CONTINUE
            PP(II,WWW)=1
            SS(II,WWW)=0
            DO 136 JJ=2,WWW
              JJJ=WWW-JJ+2
              X1(II+1,JJJ)=X1(II+1,JJJ)+PP(II,JJJ)+SS(II,JJJ)
              IF(X1(II+1,JJJ).LT.2)GO TO 136
              X1(II+1,JJJ)=X1(II+1,JJJ)-2
              SS(II,JJJ-1)=1
136         CONTINUE
C*********END OF TWO'S COMPLEMENT X1(II+1)**************
123         DO 137 JJ=1,WWW
              JJJ=WWW-JJ+1
              X1(II+1,JJJ+1)=X1(II+1,JJJ)
137         CONTINUE
            X1(II+1,1)=0
C*******THIS PART IS USED FOR TWO'S COMPLEMENT BINERY***********
C                    ADDITION
C
            DO 138 JJ=1,WW1
138           SS(II,JJ)=0
            DO 140  JJ=2,WW1
              JJJ=WW1-JJ+1
              XX(R,JJJ)=X1(II+1,JJJ)+P(II,JJJ)+SS(II,JJJ)
              IF(XX(R,JJJ).LT.2)GO TO 140
              XX(R,JJJ)=XX(R,JJJ)-2
              SS(II,JJJ-1)=1
140         CONTINUE
            IF(SS(II,1).EQ.1)GO TO 949
            IF(SS(II,2).EQ.1)GO TO 949
            DO 948 JJ=1,WWW
948           XX(R,JJ)=XX(R,JJ+1)
C
C******END OF TWO'S COMPLEMENT ADDITION***********
C
C*******THE BEGINING OF THE TWO'S COMPLEMENT OF**********
C                    SUM
C
949         IF(XX(R,1).EQ.0)GO TO 678
            DO 148 JJ=2,WWW
              IF(XX(R,JJ).EQ.0)GO TO 149
              XX(R,JJ)=0
              GO TO 148
149           XX(R,JJ)=1
```

```
148       CONTINUE
          DO 150 JJ=1,WWW-1
            PP(R,JJ)=0
            SS(R,JJ)=0
150       CONTINUE
          PP(R,WWW)=1
          SS(R,WWW)=0
          DO 151 JJ=2,WWW
            JJJ=WWW-JJ+2
            XX(R,JJJ)=XX(R,JJJ)+PP(R,JJJ)+SS(R,JJJ)
            IF(XX(R,JJJ).LT.2) GO TO 151
            XX(R,JJJ)=XX(R,JJJ)-2
            SS(R,JJJ-1)=1
151       CONTINUE
678       DO 743 JJ=1,WWW
743         X1(II+1,JJ)=XX(R,JJ)
          DO 695 JJ=1,WWW
695         XX(R,JJ)=0
C
C********END OF TWO'S COMPLEMENT OF SUM*******************
          IF(II.EQ.(R-1))GO TO 153
          GO TO 110
153       DO 610 JJ=1,WW1
            Y(R,JJ)=0
            SS(R,JJ)=0
610       CONTINUE
          DO 600 N=2,CW
            KK=CW-N+2
            IF(H(O,KK).EQ.1)GO TO 601
604         DO 602 K=2,WWW
              K1=WWW-K+2
              Y(R,K1+1)=Y(R,K1)
602         CONTINUE
            Y(R,2)=0
            GO TO 600
601         DO 603 JJ=2,WWW
              JJJ=WWW-JJ+2
              Y(R,JJJ)=Y(R,JJJ)+SS(R,JJJ)+X1(II+1,JJJ)
              IF(Y(R,JJJ).LT.2)GO TO 603
              Y(R,JJJ)=Y(R,JJJ)-2
              SS(R,JJJ-1)=1
603         CONTINUE
            IF(SS(R,1).EQ.0)GO TO 604
            DO 933 K=2,WWW
              K1=WWW-K+2
              Y(R,K1+1)=Y(R,K1)
933         CONTINUE
            Y(R,2)=1
            GO TO 604
600       CONTINUE
          DO 690 JJ=2,WWW
690         Y(R,JJ)=Y(R,JJ+1)
          IF(H(O,1).EQ.X1(II+1,1))GO TO 620
          Y(R,1)=1
          GO TO 621
620       Y(R,1)=0
621       IF(RF.NE.0)GO TO 526
          IF(R.EQ.0)GO TO 110
          WRITE(2,923)R,(Y(R,JJ),JJ=1,WWW)

       END OF CALCULATION OF EACH OUTPUT FOR FIRST
***********SECOND  ORDER FILTER**********************
```

```
            DO 888 B=F,(F+2)
              DO 777 JJ=1,WWW
                X1(B+1,JJ)=X(B+1,JJ)
 888        CONTINUE
            IF(R.EQ.(S-1))GO TO 762
            GO TO 761
 526        IF(R.EQ.0)GO TO 110
            WRITE(3,923)R,(Y(R,JJ),JJ=1,WWW)
C
C        END OF CALCULATION OF EACH OUTPUT FOR NEXT
C*********SECOND ORDER FILTER*****************************************
            DO 458 B=F,(F+2)
              DO 459 JJ=1,WWW
 459            X1(B+1,JJ)=X(B+1,JJ)
 458        CONTINUE
            IF(R.NE.(S-1))GO TO 761
            CALL CLOSE(3,IER)
            IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
 761        CONTINUE
 110      CONTINUE
C
C********END OF CALCULATION OF FIRST SECOND ORDER FILTER******
 491    CONTINUE
 40     FORMAT(12X,140(I1))
 921    FORMAT(1X,I4,3X,140(I1))
 762    CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
        CALL CLOSE (2,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
 496    DO 493 JJ=1,CW
          H(0,JJ)=H(RF+3,JJ)
          H(1,JJ)=H(RF+4,JJ)
          H(2,JJ)=H(RF+5,JJ)
 493    CONTINUE
        RF=RF+3
        IF(RF.EQ.NC)GO TO 500
        GO TO 525
 915    FORMAT(2X,I5)
 916    FORMAT(1X,I5)
C
C       END OF CALCULATION OF THE CASCADE-NESTED STRUCTURE
C             OUTPUT
C
C**********************************************************************
 500    CALL EXIT
        END
```

USER'S MANUAL PROGRAM PNES

FILE:	PNES
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Calculating the Parallel-Nested Filter Output Response.
FUNCTION:	This program is used to calculate the parallel-nested filter output response. Each second-order section is acting as an individual nested filter. The outputs of each second-order section is stored in different files. Then, they are added together in two's complement. The result will be the output response of the parallel-nested filter structure.
PROGRAM USE:	The program is loaded by the following command: RLDR PNES @FLIB@
SUBROUTINE REQUIRED:	None

FLOWGRAPH:

Type		Figure
1.	Two's Complement of Binary Numbers	26
2.	Two's Complement Addition	28
3.	Binary Multiplication	29
4.	Shift-left and Shift-right	30
5.	FIR Parallel-Nested Form Structure	36

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
PNES
NESTED STRUCTURE BINARY COEFFICIENT FILE NAME:  NC
BINARY INPUT FILE NAME:  TI
UNQUANTIZED BINARY OUTPUT NAME FOR NS:  NO
NEXT SECOND ORDER OUTPUT FILE:  NO1
```

```
FIRST SECOND ORDER FILTER OUTPUT:  NO
ENTER THE FILE NAME FOR FIRST SECOND ORDER:  NO2
NEXT SECOND ORDER OUTPUT FILE:  NO1
FIRST SECOND ORDER OUTPUT FILE:  NO2
ENTER PARALLEL OUTPUT FILE STRUCTURE:  PPO
```

The content of the file NC is the same as the file NC explained in Program CNES. The file TI is explained in Appendix B. The file NO, NO1, NO2 has the similar data to the file NO explained in Program CNES. The file PPO, representing the parallel-nested filter output response, is also similar to the file CPO explained in Program CNES.

```
C*********************************************************************************
C     PROGRAM          PNES
C     AUTHOR    .      HARUN INANLI
C     DATE             SEPTEMBER 83
C     LANGUAGE:        FORTRAN 5
C
C     FUNCTION.        THIS PROGRAM IS USED TO CALCULATE THE FILTER
C                      OUTPUT BASED ON PARALLE -NESTED STRUCTURE
C                      THAT IS, EACH SECOND ORDER COMPANENT OF THE
C                      PARALLEL FILTER ARE IN NESTED FORM THE NEGATIVE
C                      NUMBER IS REPRESENTED IN TWO'S COMPLEMENT.
C                      THEN SUMMATION IS CARRIED OUT IN THIS NUMBER
C                      SYSTEM, TOO.
C
C*********************************************************************************
      INTEGER OUTFILE(7),OUTF(7),XX(20,140),Y(20,140),X1(20,140)
      INTEGER X(20,140),H(20,140),P(20,140),S (20,140),PP(20,140)
      INTEGER IW,NC,CW,S,J,1,J1,R,K,II,JA,F,F  Q,QQ,OUTD(7),CWW
      INTEGER JB,JL,RR,OUTA(7),OUTFM(7)
      ACCEPT"NESTED STRUCTURE BINERY COEFFICIEN FILE NAME : "
      READ(11,50)OUTFILE(1)
 50   FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      READ(1,60)NC
      READ(1,60)CW
 60   FORMAT(5X,I4)
      DO 200 IJ=0,(NC-1)
        DO 201 JJ=1,(2*CW+1)
 201      H(IJ,JJ)=0
 200  CONTINUE
C***********BINERY NESTED FILTER COEFFICIENTS ARE READ BY********
C                   MEANS OF CHANNEL(1)
C
      DO 70 I=0,(NC-1)
 70     READ(1,80)(Q,(H(I,K),K=1,CW))
 80   FORMAT(1X,I4,10X,140(I1))
      CALL CLOSE(1,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C
C************NESTED FILTER COEFFICIENT***********************
C
C*********THE INPUT TO THE FILTER IS READ FROM*****************
C                 THE FILE BY MEANS OF CHANNEL(1)
C
      ACCEPT"BINERY INPUT FILE NAME : "
      READ(11,10)OUTFILE(1)
 10   FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER.NE.1)TYPE"OPEN INPUT FILE ERROR   ",IER
      READ(1,30) S
 30   FORMAT(20X,I5)
      READ(1,30)IW
```

```
C*********FIRST SECOND ORDER FILTER OUTPUT IS STORED*************
                 IN THE FILE BY MEANS OF CHANNEL (2)
C
      ACCEPT"UNQUANTIZED BINERY OUTPUT NAME FOR NS : "
      READ(11,100)OUTF(1)
100   FORMAT(S15)
      CALL DFILW(OUTF,IER)
      IF(IER.EQ.13)GO TO 101
      IF(IER.NE.1)TYPE"DELETE FILE ERROR",IER
101   CALL CFILW(OUTF,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      WW=2*IW
      WWW=2*IW+1
      IWW=IW+1
      WW1=2*IW+2
      CWW=CW+1
C*************************************************************
C
C     THE BEGINING OF THE CALCULATION OF THE OUTPUT FOR
C              EACH SECOND ORDER NESTED STRUCTURE
C
      RF=0
      QQ=0
525   IB=0
      IA=0
      IC=0
      R=0
      IF(RF.EQ.0)GO TO 513
      IF(RF.GT.(NC-1))GO TO 500
C******NEXT SECOND ORDER OUTPUT IS WRITTEN TO THE FILE***********
                 BY MEANS OF CHANNEL(3)
C
      ACCEPT"NEXT SECOND ORDER OUTPUT FILE : "
      READ(11,100)OUTD(1)
      CALL DFILW(OUTD,IER)
      IF(IER.EQ.13)GO TO 584
      IF(IER.NE.1)TYPE "DELETE FILE ERROR",IE
584   CALL CFILW(OUTD,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(3,OUTD,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
      REWIND 1
      READ(1,30)S
      READ(1,30)IW
513   CONTINUE
400   I=R
      IF(I.EQ.360)GO TO 434
      IF(I.EQ.300)GO TO 434
      IF(I.EQ.240)GO TO 434
      IF(I.EQ.180)GO TO 434
      IF(I.EQ.120)GO TO 434
      IF(I.EQ.60)GO TO 434
      IF(IA.EQ.360)GO TO 433
      IF(IA.EQ.300)GO TO 433
      IF(IA.EQ.240)GO TO 433
      IF(IA.EQ.180)GO TO 433
      IF(IA.EQ.120)GO TO 433
      IF(IA.EQ.60)GO TO 433
```

FOLLOWING

PAGES

```
C     ****THE LOOP 20 IS USED TO READ THE INPUT OF THE FILTER********
      DO 20 J=IA,(IA+9)
        DO 23 JJ=1,IW
          X(J,JJ)=0
        READ(1,40,END=41)(X(J,KK),KK=1,IW)
      CONTINUE
C********END OF LOOP 20*****************************************
C     IF(IB.EQ.1)GO TO 412
C    ****THIS PART OF THE PROGRAM IS USED TO**   ********
C                FIND THE Y(0)

      IB=1
      DO 255 JJ=1,WWW,WWW
        X(0,JJ)=0
      DO 413 JJ=1,WWW
        Y(0,JJ)=0
        SS(0,JJ)=0
413   CONTINUE
      DO 414 N=2,CW
        KK=CW-N+2
        IF(H(0,KK).EQ.1)GO TO 415
        DO 416 K=2,WWW
          K1=WWW-K+2
          Y(0,K1+1)=Y(0,K1)
416     CONTINUE
        Y(0,2)=0
        GO TO 414
        DO 417 JJ=2,WWW
          JJJ=WWW-JJ+2
          Y(0,JJJ)=Y(0,JJJ)+SS(0,JJJ)+X(0,JJJ)
          IF(Y(0,JJJ).LT.2)GO TO 417
          Y(0,JJJ)=Y(0,JJJ)-2
          SS(0,JJJ-1)=1
417     CONTINUE
        IF(SS(0,1).EQ.0)GO TO 418
        DO 417 K=2,WWW
          K1=WWW-K+2
          Y(0,K1+1)=Y(0,K1)
        CONTINUE
        Y(0,2)=1
        GO TO 418
      CONTINUE
      DO 333 JJ=1,WWW
        Y(0,JJ)=Y(0,JJ+1)
      IF(RF.NE.0)GO TO 455
      WRITE(2,923)QQ,(Y(0,JJ),JJ=1,WWW)
      GO TO 412
      WRITE(3,923)QQ,(Y(0,JJ),JJ=1,WWW)
C
C     ****THE COMPLETION OF Y(0)****************   ********
```

```
      (A=J
      IF(RF EQ [illegible] TO 50[illegible]
      DO 401 R=[illegible]
        IF(P EQ [illegible] TO 4[illegible]
        DO 501 L=1,WWL
          [illegible]X(R,I)=0
        IF R GT [illegible] TO 310
        KKK=R
        F=0
        GO TO 31[illegible]
        KKK=2
        F=R-2
        DO 355 JJ=1WW,WWW
          X(R,JJ)=0
        DO 778 JJ=1,WWW
          X1(R,JJ)=X(R,JJ)
        IF(R GE (I+9))GO TO 400
C*****THE LOOP 110 IS USED TO CALCULATE THE OUTPUT*************
              OF EACH SECOND ORDER FILTER ONE [illegible] ONE

        DO 110 II=F,(F+2)
          IF(KKK GE 2)GO TO 444
          J1=KKK-II
          GO TO 44[illegible]
          J1=R-II
          IF(J1.LE 0)GO TO 401
          DO 560 JJ=IWW,WWW
            H(J1,JJ)=0
          DO 111 JJ=1,WWW
            SS(II,JJ)=0
            P(II,JJ)=0
111       CONTINUE
C**********THE LOOP 112 IS USED FOR BINARY MULTIPLICATION***********
          DO 112 N=2,WWW
            KK=WWW-N+2
            IF(H(J1,KK).EQ.1)GO TO 113
            DO 114 K=2,WWW
              K1=WWW-K+2
              P(II,K1+1)=P(II,K1)
114         CONTINUE
            P(II,2)=0
            GO TO 112
113         DO 115 JJ=2,WWW
              JJJ=WWW-JJ+2
              P(II,JJJ)=P(II,JJJ)+X1(II,JJJ)+SS(II,JJJ)
              IF(P(II,JJJ).LT.2)GO TO 115
              P(II,JJJ)=P(II,JJJ)-2
              SS(II,JJJ-1)=1
115         CONTINUE
            IF(SS(II,1).EQ.0)GO TO 116
            DO 760 K=2,WWW
              K1=WWW-K+2
              P(II,K1+1)=P(II,K1)
760         CONTINUE
            P(II,2)=1
            GO TO 116
116       CONTINUE
C**********END OF LOOP 112**************************
```

```
      DO 580 JJ=2,WWW
        P(II,JJ)=P(II,JJ+1)
      IF(H(JI,1).EQ.X1(II,1))GO TO 118
      P(II,1)=1
      GO TO 119
      P(II,1)=0
C     ******THE BEGINING OF THE TWO'S COMPLEMENT OF P********

      IF(P(II,1).EQ.0)GO TO 120
      DO 121 JJ=2,WWW
        IF(P(II,JJ).EQ.0)GO TO 122
        P(II,JJ)=0
        GO TO 121
        P(II,JJ)=1
      CONTINUE
      DO 130 JJ=1,WWW-1
        PP(II,JJ)=0
        SS(II,JJ)=0
      CONTINUE
      PP(II,WWW)=1
      SS(II,WWW)=0
      DO 131 JJ=2,WWW
        JJJ=WWW-JJ+2
        P(II,JJJ)=P(II,JJJ)+PP(II,JJJ)+SS(II,JJJ)
        IF(P(II,JJJ).LT.2)GO TO 131
        P(II,JJJ)=P(II,JJJ)-2
        SS(II,JJJ-1)=1
      CONTINUE

C     ******END OF TWO'S COMPLEMENT OF P*********

C     ******THE BEGINING OF TWO'S COMPLEMENT OF X1(II+1)********

      IF(X1(II+1,1).EQ.0)GO TO 123
      DO 124 JJ=2,WWW
        IF(X1(II+1,JJ).EQ.0)GO TO 126
        X1(II+1,JJ)=0
        GO TO 124
        X1(II+1,JJ)=1
      CONTINUE
      DO 135 JJ=1,WWW-1
        PP(II,JJ)=0
        SS(II,JJ)=0
      CONTINUE
      PP(II,WWW)=1
      SS(II,WWW)=0
      DO 136 JJ=2,WWW
        JJJ=WWW-JJ+2
        X1(II+1,JJJ)=X1(II+1,JJJ)+PP(II,JJJ)+SS(II,JJJ)
        IF(X1(II+1,JJJ).LT.2)GO TO 136
        X1(II+1,JJJ)=X1(II+1,JJJ)-2
        SS(II,JJJ-1)=1
      CONTINUE

C     ******THE COMPLETION OF TWO'S COMPLEMENT OF X1(II+1)*******
      DO 137 JJ=1,WWW
        JJJ=WWW-JJ+1
        X1(II+1,JJJ+1)=X1(II+1,JJJ)
      CONTINUE
      X1(II+1,1)=0
C     ******TWO'S COMPLEMENT BINERY ADDITION***********
```

```
      DO 140 JJ=1,WW1
        [illegible]
      DO 140 JJ=2,WW1
        JJJ=[illegible]-JJ+1
        XX(R,JJJ)=X1(II+1,JJJ)+P(II,JJJ)   S(II,JJJ)
        IF(XX(R,JJJ).LT.2)GO TO 140
        XX(R,JJJ)=XX(R,JJJ)-2
        SS(R,JJJ-1)=1
      CONTINUE
      IF(SS(1,1).EQ.1)GO TO 949
      IF(SS(1,2).EQ.1)GO TO 949
      DO 748 JJ=1,WWW
        XX(R,JJ)=XX(R,JJ+1)
      IF(XX(R,1).EQ.0)GO TO 678
      DO 148 JJ=2,WWW
        IF(XX(R,JJ).EQ.0)GO TO 149
        XX(R,JJ)=0
        GO TO 148
        XX(R,JJ)=1
      CONTINUE
      DO [illegible] =1,WWW-1
        [illegible]=0
        [illegible]=0
      CONTINUE
      PP(R,[illegible])=1
      SS(R,[illegible])=0
      DO 151 [illegible]=2,WWW
        JJJ=[illegible]-JJ+2
        XX(R,JJJ)=XX(R,JJJ)+PP(R,JJJ)+SS(R,JJJ)
        IF(XX(R,JJJ).LT.2) GO TO 151
        XX(R,JJJ)=XX(R,JJJ)-2
        SS(R,JJJ-1)=1
151   CONTINUE
C*******COMPLETITION OF ADITTION*****************    ****
      DO 743 JJ=1,WWW
743     X1(II+1,JJ)=XX(R,JJ)
      DO 695 JJ=1,WWW
        XX(R,JJ)=0
      IF(II.[illegible](R-1))GO TO 153
      GO TO 110
      DO 610 JJ=1,WW1
        Y(R,JJ)=0
        SS(R,JJ)=0
610   CONTINUE
      DO 600 N=2,CW
        KK=CW-N+2
        IF(H(Q,KK).EQ.1)GO TO 601
        DO 602 K=2,WWW
          K1=WW9-K+2
          Y(R,K1+1)=Y(R,K1)
        [illegible] END
```

```
          Y(R, ...)
          GO TO ...
          DO ... JJ=2,WWW
            ...WWW-JJ+2
            Y(R,...JJ)=Y(R,JJJ)+ABS(R,JJJ)*X1(I...,JJJ)
            IF(Y(R,JJJ).LT.2)GO TO 603
            Y(R,JJJ)=Y(R,JJJ)-2
            Y(R,JJJ-1)=1
          CONTINUE
          IF(Y(R,1).EQ.0)GO TO 604
          DO 702 K=2,WWW
            K1=WWW-K+2
            Y(R,K1+1)=Y(R,K1)
          CONTINUE
          Y(R,2)=1
          GO TO 604
        CONTINUE
        DO 620 JJ=2,WWW
          Y(R,JJ)=Y(R,JJ+1)
        IF(H(... .EQ.X1(I+1,1))GO TO 620
        ...(R,1)=...
        GO TO ...
        ...
        IF(RE...)GO TO 526
        IF(R.EQ. ...)GO TO 110
        WRITE(2,...)R,(Y(R,JJ),JJ=1,WWW)
C *****FIRST SECOND ORDER SECTION OUTPUT IS WRITTEN TO THE FILE*****
        DO 888 B=F,(F+2)
          DO 777 JJ=1,WWW
            X1(B+1,JJ)=X(B+1,JJ)
        CONTINUE
        IF(R.EQ.(S-1))GO TO 762
        GO TO 761
  ...   IF(R.EQ.0)GO TO 110
        WRITE(3,...3)R,(Y(R,JJ),JJ=1,WWW)
C
C *****NEXT SECOND ORDER SECTION IS WRITTEN TO THE FILE*******
        DO 450 B=F,(F+2)
          DO 457 JJ=1,WWW
            X1(B+1,JJ)=X(B+1,JJ)
        CONTINUE
        IF(R.NE.(S-1))GO TO 761
        CALL CLOSE(3,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
        CALL CLOSE(1,IER)
        IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
        CONTINUE
      CONTINUE
     CONTINUE
     FORMAT(12X,140(I1))
     FORMAT(1X,I4,...X,140(I1))
 ... CALL CLOSE(...,IER)
     IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
```

```
      DO ... ... W
        ... ...+3, ... )
        ... ...+4 ... )
        ... ...+5 ... )
      CONTINUE
      RR=RR+3
      IF(... EQ NC)GO TO 196
      GO TO 525
C*******FIRST SECOND ORDER OUTPUT IS READ BY*********
C             MEANS OF CHANNEL(2)

      ACCEPT"FIRST SECOND ORDER FILTER OUTPUT     "
      READ(11,100)OUTF(1)
      CALL OPEN(2,OUTF,1,IER)
      IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
      REWIND 2
C*******PARALEL-NESTRED FILTER OUTPUT IS WRITTEN*********
C             TO THE FILE AFTER ADDITION OF OL...
C             OUTPUT AND FIRST SECOND ORDER SECTION
C             OUTPUT

      ACCEPT"ENTER THE FILE NAME FOR FIRST SECOND ORDER : "
      READ(11,100)OUTFM(1)
      CALL DFILW(OUTFM,IER)
      IF(IER.EQ.1)GO TO 386
      IF(IER.NE.1)TYPE"DELETE FILE ERROR ",IER
      CALL CFILW(OUTFM,2,IER)
      IF(IER.NE.1)TYPE "CREATE FILE ERROR ",IE
      CALL OPEN(6,OUTFM,3,IER)
      IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
      GG=0
      J=0
      JA=0
      RR=0
      JB=0
      IF(JB.EQ.0)GO TO 354
      JB=J+1
      RR=JB
      IF(GG.NE.0)GO TO 316
C*******LOOP 192 IS USED TO READ THE FIRST SECOND ORDER*********
C             SECTION OUTPUT

      DO 192 JA=RR,(RR+9)
        DO 213 JJ=1,WWW
          X(JA,JJ)=0
        READ(2,923,END=193,ERR=500)J,(X(JA,K5),K5=1,WWW)
      CONTINUE
      CONTINUE

C*******END OF LOOP 192*************************************
```

```
      DO 314 JE=JB,(JB+9)
        DO 315 JJ=1,WWW
          ...(JE,JJ)=0
          ...(JE,JJ)=0
        CONTINUE
      CONTINUE
      GO TO 313
      IF(J.GE.9)GO TO 364
C*****NEXT SECOND ORDER OUTPUT IS READ BY MEANS OF*****
C              CHANNEL (3)

      ACCEPT "NEXT SECOND ORDER OUTPUT FILE :
      READ(11,100)OUTD(1)
      CALL OPEN(3,OUTD,1,IER)
      IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
      REWIND 3
C********FIRST SECOND ORDER OUTPUT IS READ BY MEANS OF********
C              CHANNEL(6)

      ACCEPT "FIRST SECOND ORDER OUTPUT FILE : "
      READ(11,100)OUTFM(1)
      CALL OPEN(6,OUTFM,1,IER)
      IF(IER.NE.1)TYPE "OPEN FILE ERROR",IER
      REWIND 6
C********PARALEL-NESTED FILTER STRUCTURE OUTPUT IS *************
C              WRITTEN BY MEANS OF CHANNEL(5)

      ACCEPT"ENTER PARALEL OUTPUT FILE STRUCTURE : "
      READ(11,100)OUTA(1)
      CALL DFILW(OUTA,IER)
      IF(IER.EQ.13)GO TO 365
      IF(IER.NE.1)TYPE "DELETE FILE ERROR",IER
      CALL CFILW(OUTA,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN (5,OUTA,3,IER)
      IF(IER.NE.1)TYPE"OPEN FILE ERROR",IER
C*****LOOP 323 IS USED TO READ THE OUTPUT OF THE FIRST*******
C              AND SECOND ORDER SECTION

      DO 323 JA=IR,(IR+9)
        DO 366 JJ=1,WWW
          X(JA,JJ)=0
        IF(JA.GT.(5.1))GO TO 500
        READ(3,923,END=324,ERR=500)J,(X(JA,K9),K9=1,WWW)
        READ(6,923,END=324,ERR=500)J,(Y(JA,KK5),KK5=1,WWW)
      CONTINUE
      CONTINUE

C**END OF LOOP 323***********************
      DO 314 J=JB,(JB+9)
        DO 315 JJ=1,WWW
          ...(J,JJ)=0
      CONTINUE
```

```
****TWO'S COMPLEMENT ADDITION OF FIRST AND N  1 SECOND*********
              ORDER SECTION OUTPUT

      DO 194 J=JB,(JB+9)
        DO 195 K=2,WWW
          JJ=WWW-K+1
          Y(J,JJ)=Y(J,JJ)+X(J,JJ)+SS(J,JJ)
          IF(Y(J,JJ).LT.2)GO TO 195
          Y(J,JJ)=Y(J,JJ)-2
          SS(J,JJ-1)=1
        CONTINUE
        IF(SS(J,1).EQ.1)GO TO 216
        IF(SS(J,2).EQ.1)GO TO 216
        GO TO 217

    ***END OF ADDITION*********************
        DO 218 JJ=1,WWW
          II=WWW-JJ+1
          Y(J,II)=Y(J,II)
        CONTINUE
        IF(QG.EQ.0)GO TO 369
        WRITE(5,923)J,(Y(J,JJ),JJ=1,WWW)

   ***PARALLEL-NESTED FILTER OUTPUT IS WRITTEN TO THE FILE********
        GO TO 308
        WRITE(6,923)J,(Y(J,JJ),JJ=1,WWW)

   ****FIRST SECOND ORDER SECTION IS WRITTEN TO THE FILE******
        IF(J.GE.(S-1))GO TO 311
        IF(J.GE.(JB+9))GO TO 221
      CONTINUE
      QG=QG+1
      J=-1
      JB=0
      JA=0
      CALL CLOSE(6,IER)
      IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
      IF(QG.GE.2)GO TO 373
      GO TO 322
      CALL CLOSE(5,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
      CALL EXIT
      END
```

Appendix D

Digital Filter Outputs and Plots

Appendix D contains the program and user's manual for digital filter outputs and plots. Each program user's manual explains what the program does. These are called as follows:

1. OUT1
2. PLOT
3. PLOT1

USER'S MANUAL PROGRAM OUT1

FILE:	OUT1
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Quantizing the Unquantized Output.
FUNCTION:	This program quantizes the output filter response according to user requirements of either the truncating or the rounding technique.
PROGRAM USE:	The program is loaded by the following command: RLDR OUT1 @FLIB@
SUBROUTINE REQUIRED:	None

FLOWGRAPH:

Type	Figure
1. Two's Complement of Binary Numbers	26
2. Binary to Decimal Converter	27

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
OUT1
ENTER UNQUANTIZE OUTPUT FILE NAME:  NO
ENTER OUTPUT FILE NAME FOR PLOT:  PO
QUANTIZATION TYPE (1-TRUNCATION, 0-ROUNDING) 1
```

The file NO, representing the digital filter output in binary, is explained in Appendix C. The file PO shown below is representing the number of coefficient with 100 at the top, the coefficient numbers at the first column, the

truncated coefficients based on 20 bits output register at the second column, the truncated coefficients based on 10 bits output register at the third column and the difference between these two truncated coefficients.

PO

100

0	.9727478E-03	.0000000E 00	.9727478E-03
1	.3112793E-02	.1953125E-02	.1159668E-02
2	.6874084E-02	.5859375E-02	.1014709E-02
3	.9014130E-02	.7812500E-02	.1201630E-02
4	.9986877E-02	.9765625E-02	.2212524E-03
5	.9986877E-02	.9765625E-02	.2212524E-03
6	.9986877E-02	.9765625E-02	.2212524E-03
7	.9986877E-02	.9765625E-02	.2212524E-03
8	.9986877E-02	.9765625E-02	.2212524E-03
9	.9986877E-02	.9765625E-02	.2212524E-03
10	.9014130E-02	.7812500E-02	.1201630E-02
11	.6874084E-02	.5859375E-02	.1014709E-02
12	.3112793E-02	.1953125E-02	.1159668E-02
13	.9727478E-03	.0000000E 00	.9727478E-03
14	.0000000E 00	.0000000E 00	.0000000E 00
15	.0000000E 00	.0000000E 00	.0000000E 00
16	.0000000E 00	.0000000E 00	.0000000E 00
17	.0000000E 00	.0000000E 00	.0000000E 00
18	.0000000E 00	.0000000E 00	.0000000E 00
19	.0000000E 00	.0000000E 00	.0000000E 00
20	.0000000E 00	.0000000E 00	.0000000E 00

```
C****    ****************************************    *****************
C
C        PROGRAM              QUT1
C        AUTHOR               HARUN INANLI
C        DATE                 LANGUAGE
C
C        FUNCTION             THIS PROGRAM CONVERTS THE   BINARY REPRESENTATION
C                             OF THE DICITEL FILTER OUT   T RESPONSE TO THE
C                             DECIMEL NUMBER SYSTEM.
C
C******   ****************************************    ******************
      DIMENSION YY(500),YT(500),D(500)
      INTEGER OUTFILE(7),OPT,MM(20,140),SS(20,1
      INTEGER Y(20,140),M(20,140),OUTF(5)
      INTEGER W,OW,S,RR
      ACCEPT "ENTER UNQUANTIZED OUTPUT FILE NAM      "
      READ(11,10)OUTFILE(1)
   10 FORMAT(S15)
      CALL OPEN(1,OUTFILE,1,IER)
      IF(IER NE 1)TYPE"OPEN FILE ERROR",IER
      READ(1,20)OW
   20 FORMAT(2X,I5)
      READ(1,30)S
   30 FORMAT(1X,I5)
   50 FORMAT(1X,I4,3X,140(I1))
      ACCEPT"ENTER OUTPUT FILE NAME FOR PLOT
      READ(11,900)OUTF(1)
  900 FORMAT(S15)
      CALL CFILW(OUTF,IER)
      IF(IER EQ 13)GO TO 910
      IF(IER NE.1)TYPE"DELETE FILE ERROR",IER
  910 CALL CFILW(OUTF,2,IER)
      IF(IER.NE.1)TYPE"CREATE FILE ERROR",IER
      CALL OPEN(2,OUTF,3,IER)
      IF(IER NE.1)TYPE"OPEN FILE ERROR",IER
      ACCEPT"QUANTIZATION TYPE(1-TRUNCATION,0-R    DING)",OPT
      WRITE(2,231)S
  231 FORMAT(20X,I5)
      DO 40 RR=0,(S-1),20
        TYPE RR
        DO 41 I=RR,(RR+19)
          READ(1,50,END=41)Q,(Y(I,K),K=1,2*OW+1
          IF(OPT.EQ 0)GO TO 300
          YY(I)=0.0
C****    ****************************************    **********
C
C        TRUNCATION OPTION
C
```

```
C****      THE LOOP 80            CULATE********          ******
C                 THE QUANTIZED OUTPUT
C
                  DO  II=2,2*OW
                    YY(I)=YY(I)+Y(I,II)*(2.0**(-II+1)
                  IF(Y(I,1) .EQ.0)GO TO 90
                  YY(I)=-YY(I)
C
C*****  END OF LOOP 80****************************
   90             YT(I)=0.0
C*******THE LOOP 140 IS USED TO CALCULATE *****
C                 THE QUANTIZE OUTPUT
C
                  DO 140 I=2,OW
   140              YT(I)=YT(I)+    (I)*(2.0**(-II+1
                  IF(Y(I,1) .EQ.0)GO TO 150
                  YT(I)=-YT(I)
C
C*****   ***END OF LOOP 140***********************
   150            D(I)=YY(I)-YT(I)
C*****   THE PART OF TRUNCATION IS USED TO WRITE       ****
C                 THE INFORMATION OBTAINED ABOVE
C                 TO THE FILE
                  WRITE(2,201)I,YY(I),YT(I),D(I)
   201            FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E1
                  GO TO 310
C
C          END OF TRUNCATION
C
C****     ******************************************     ***
C*****    ******************************************     ****
C
C          ROUNDING OPTION
C
   300            YY(I)=0.0
                  D(I)=0.0
                  DO 321 K=1,(OW+1)
   321              SS(I,K)=0
                  DO 350 K=1,(OW+1)
   350              M(I,K)=0
                  M(I,OW)=1
                  NNN=OW+1
                  IF(Y(I,NNN).EQ.0)GO TO 360
                  DO 370 JJ=3,NNN
                    II=NNN-JJ+2
                    MM(I,II)=Y(I,II)+M(I,II)+SS(I,II)
                    IF(MM(I,II).LT.2)GO TO 370
                    MM(I,II)=MM(I,II)-2
                    SS(I,II-1)=1
   370            CONTINUE
                  GO TO 361
   360            DO 390 K=2,OW
   390              MM(I,K)=Y(I,K)
   361            MM(I,1)=Y(I,1)
```

```
C***     THE LOOP 400 IS USED TO FIND THE QUANTIZ[illegible]    ****
C                 OUTPUT
C
            DO 400 K=2,OW
   400        YY(I)=YY(I)+MM(I,K)*(2.0**(-K+1))
C
C*****   END OF LOOP 400************
            IF(MM(I,1).EQ.0)GO TO 410
            YY(I)=-YY(I)
   410      YT(I)=0.0
C*****   THE LOOP 440 IS USED TO FIND THE UNQUANTI[illegible]  *****
C                 OUTPUT
C
            DO 440 II=2,2*OW+1
   440        YT(I)=YT(I)+Y(I,II)*(2.0**(-II+1))
C
C****     END OF LOOP 440********
            IF(Y(I,1).EQ.0)GO TO 450
            YT(I)=-YT(I)
   450      D(I)=YT(I)-YY(I)
C*************THIS PART OF THE ROUNDING OPERATION [illegible]  ******************
C                 USED TO WRITE THE INFORMATION OB[illegible]NED
C                 ABOVE TO THE FILE
            WRITE(2,501)I,YT(I),YY(I),D(I)
   501      FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E1[illegible]
   [illegible]        CONTINUE
   [illegible]      CONTINUE
   4[illegible]    CONTINUE
C
C        END OF ROUNDING OPTION
C
C*****************************************************************  ****
4[illegible]       CALL CLOSE(1,IER)
         IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
         CALL CLOSE(2,IER)
         IF(IER.NE.1)TYPE "CLOSE FILE ERROR",IER
         STOP
         END
```

USER'S MANUAL PROGRAM PLOT

FILE:	PLOT
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Producing the Input Signal Plot.
FUNCTION:	This program plots both the input and the scaled, as well as the quantized, input signals. These data come from the file TI1.
PROGRAM USE:	The program is loaded by the following command:
	RLDR PLOT GRPH.LB @FLIB@

SUBROUTINE REQUIRED:

Name	Location	Purpose
GRPH.LB	DP4F	General graph plot

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
PLOT
INPUT FILE ANME FOR PLOT:  TI1
```

The content of the file TI1 is explained in Appendix B.

```
C*****************************************************************************
C
C
C     AUTHOR        HAKUN INANC
C     DATE          SEPTEMBER 83
C     LANGUAGE :    FORTRAN 5
C
C     FUNCTION .    THE PROGRAM PLOTS BOTH THE ... UT AND
C                   THE SCALED AS WELL AS THE ...TIZED INPUT
C                   SIGNALS
C
C*****************************************************************************
            OUTFILE(7), ... NG.N
      DIMENSION X(500),XS(500),B(500),T(500),...
            XX(500)
            INPUT FILE NAME FOR PLOT : "
            ...OUTFILE(1)
 1    ...
            ...OUTFILE, ..., IER
            ...
 2    ...
      DO 30 I=1,...
      READ(...)I,X(I),XS(I),...(I)
 4... FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E14.7)
      CALL CLOSE(1,IER)
      IF(IER.NE.1)TYPE"CLOSE FILE ERROR",IER
C*****************************************************************************
C
C     PLOT FOR INPUT
C
      DO ... I=1,...
            ...X(I)
            ...X(I)
 ...  CONTINUE
      ...
      ...
      ...SCL ...
      CALL ...(“INPUT PLOT",NG,T,U,N,MODE,YMI...X,IFSCL)
      CALL ...CHAR(ICHAR,IER)
      IF(IER.NE.1)TYPE"CONTINUE CHARACTER ERROR" ...
C
C     END OF INPUT
C
C*****************************************************************************
```

```
C     **************************************************      *****************
C
C     [illegible] QUANTIZED [illegible]
C
      DO [illegible] I=1,G
      [illegible] (I)
      [illegible] (I)
      [illegible]
      CALL [illegible] [illegible] INP[illegible]",NG,T,U,[illegible] [illegible]MIN,YMAX,IFSCL)
      [illegible]
      [illegible] CHARECTER ERRO[illegible] [illegible]R
C
C     END OF QUANTIZED INPUT
C
C*    **************************************************      *****************
C**   **************************************************      **********
C
C     [illegible] SCALED INPUT
C
      DO [illegible] I=1,[illegible]
      [illegible]
      [illegible]
      [illegible]
      CALL [illegible]"SCALED INPUT",NG,T,U,N,MODE [illegible] [illegible]MAX,IFSCL)
C
C     [illegible] SCALED INPUT
C
C**   **************************************************      *****************
      [illegible]
      [illegible]
```

USER'S MANUAL PROGRAM PLOT1

FILE:	PLOT1
DIRECTORY:	DP4:OWEN
LANGUAGE:	FORTRAN 5
DATE:	September 1983
AUTHOR:	Harun Inanli
SUBJECT:	Producing the Output Response Plot.
FUNCTION:	This program plots the output response of the digital filter according to data given by the file PO. The contents of the file PO is explained in Program OUT1.
PROGRAM USE:	The program is loaded by the following command.
	RLDR PLOT1 GRPH.LB @FLIB@

SUBROUTINE REQUIRED:

Name	Location	Purpose
GRPH.LB	DP4F	General graph plot

EXECUTION OF THE PROGRAM AND ITS RESULTS:

```
PLOT1
QUANTIZE OUTPUT FILE NAME FOR PLOT:  PO
```

The contents of the file PO is explained in Program OUT1.

```
C*****************************************************************************
C
C                           PLOT
C     AUTHOR                HAKAN ...
C     DATE                  SEPTEMBER 83
C     LANGUAGE              FORTRAN 5
C
C     ABSTRACT ON           THE PROGRAM PLOTS THE OUT      BASED ON THE
C                           OUTPUT REGISTER WORD LENG      WHICH IS EQUAL
C                           AND TWO TIMES LARGER THEN      OUT WORD LENGTH
C                           AND THE DIFFERENT BETWEEN      THESE TWO OUTPUTS
C
C*****************************************************************************
      INTEGER OUTFILE(7),S,I,NCHN
      DIMENSION X(500),XS(500),D(500),T(500)
      DIMENSION XX(  )
      TYPE "QUANTIZED OUTPUT FILE NAME FOR PL
      READ(11,10)OUTFILE(1)
   10 FORMAT(S13)
      CALL OPEN(1,OUTFILE,  IER)
      ...  20)S
      ...
      DO   I=0,(S-1)
      READ(1,40,END=41) I,X(I),XS(I),D(I)
      ...   I,X(I),XS(I),D(I))
      ...
      ...
   40 FORMAT(1X,I4,2X,E14.7,2X,E14.7,2X,E14.7)
      CALL CLOSE(1,IER)
      IF ...  NE 1) TYPE"CLOSE FILE ERROR",IER
C*****************************************************************************
C
C     PLOT FOR OUTPUT USING THE LARGER WORD LE
C
      ...
      DO    I=0,(S-1)
      T   X(I)
          Y(I)
      ...
      ...
      ...
      ...
      CALL      ...    N,MODE,YHIG...     ...FSCL)
      ...
      ...                          CHARACTER ERR        R
C
C     ...      PLOT
C
C*****************************************************************************
```

```
C      ***********************                **************
C
C      OR OUTPUT ... CHARACTER WORD ...
C
       ...
       ...X5(I)
       ...Y5(I)
       ...NE
       ...GRAPH2(...PUT...,NG,...,MODE,YMIN...      IFSCL)
       ...CHAR(I...
       ...NE.1)TYPE'CONTINUE CHARECTER ERR...ER
C
C      ...PLOT
C
C      ***********************                ******************
       ...
       ...
       ...D(I)
       ...NE
       ...GRAPH2(...ROR A... THE OUTPUT",NG,T...    ...,YMIN,YMAX,IFSCL)
```

Copy available to DTIC does not permit fully legible reproduction

Bibliography

1. Oppenheim, A. V. and R. W. Schafer. Digital Signal Processing. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.

2. Shannon, C. E. "A Mathematical Theory of Communication," Bell System Tech. J., 27: 379-423 (July 1948).

3. Kreyzig, Erwin. Advanced Engineering Mathematics (Fourth Edition). New York, Chichester, Brisbane, Toronto: John Wiley and Sons, 1979.

4. Jury, E. I. Theory and Application of the Z-Transform Method. New York: John Wiley and Sons, 1964.

5. D'Azzo, J. J. and C. H. Houpis. Linear Control System Analysis and Design (Second Edition). New York: McGraw-Hill Book Company, 1981.

6. Rabiner, L. R., J. F. Kaiser, O. Herrmann, and M. T. Dolan. "Some Comparisons Between FIR and IIR Digital Filters," Bell System Tech. J., 53: 305-331 (February 1974).

7. Mano, M. Morris. Digital Logic and Computer Design. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

8. Baer, Jean-Loup. Computer Systems Architecture. Computer Science Press, 1980.

9. Hwang, K. Computer Arithmetic, Principles and Design. New York: John Wiley, 1979.

10. Antoniou, Andreas. Digital Filters: Analysis and Design. McGraw-Hill Book Company, 1979.

11. Kaiser, J. F. "Some Practical Considerations in the Realization of Linear Digital Filters," Proc. 3rd Annual Allerton Conf. on Circuit and System Theory, 621-633 (1965).

12. Knowles, J. B. and E. M. Olcayto. "Coefficient Accuracy and Digital Filter Response," IEEE Transactions on Circuit Theory, CT-15: 31-41 (March 1968).

13. Mahanta, A., R. C. Agarwal and S. C. Dutta Roy. "FIR Filter Structures Having Low Sensitivity and Roundoff Noise," IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-30: 913-919 (December 1982).

14. Jacson, L. B., S. F. Kaiser and Henry S. McDonald. "An Approach to the Implementation of Digital Filters," IEEE Transactions on Audio and Electroacoustics, AU-16: 413-421 (September 1968).

VITA

Harun Inanli was born 1 January 1956 in Fatsa, Turkey. He graduated from Air Force High School, Ciğli, Turkey, in 1976. He then attended the Air Force Academy in Istanbul, Turkey, where he received the Bachelor of Science in Electrical Engineering degree in 1978. From there he went to Flying School, Ciğli, Turkey, for five months, and Missile Training School, Gaziemir, Turkey, for eight months and was assigned to 15th Missile Base, Alemdağ, Turkey, as a Firing Control Officer.

He attended Turkish Air Force Language School in 1980 to learn English for six months before he entered the Air Force Institute of Technology.

Permanent Address: Eski Belediye cad. No. 6/A
Fatsa, ORDU, TURKEY

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b. RESTRICTIVE MARKINGS:

2a. SECURITY CLASSIFICATION AUTHORITY:

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE:

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GE/EE/83D-32

5. MONITORING ORGANIZATION REPORT NUMBER(S):

6a. NAME OF PERFORMING ORGANIZATION: School of Engineering

6b. OFFICE SYMBOL (If applicable): AFIT/ENG

7a. NAME OF MONITORING ORGANIZATION:

6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433

7b. ADDRESS (City, State and ZIP Code):

8a. NAME OF FUNDING/SPONSORING ORGANIZATION:

8b. OFFICE SYMBOL (If applicable):

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c. ADDRESS (City, State and ZIP Code):

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE (Include Security Classification): See Box 19

12. PERSONAL AUTHOR(S): Harun Inanli, 1st Lt, Turkish Air Force

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ____ TO ____	1983 December	223

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
Lynn Wolaver 7 Feb 84
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	04	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: STUDY OF FINITE WORD LENGTH EFFECTS IN SOME SPECIAL CLASSES OF DIGITAL FILTERS

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Dr Vaqar Syed, AFIT/WPAFB	255-3576	AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

One of the main problems in digital filter implementation is that all practical devices are of finite precision. Therefore, the finite word length effect of digital filters is an area of high interest.

There are various types of digital filter structures. Due to the effect of finite word length registers, each digital filter structure gives a slightly different output response for the same transfer function. Therefore, it is important to find the best filter structure which has the lowest affect on the output response for the same transfer function.

In this paper, six IIR (Infinite Impulse Response) digital filters and six FIR (Finite Impulse Response) digital filters are investigated, theoretically, for the low sensitivity due to a finite word length register. In addition, the six FIR digital filters are simulated by computer to obtain practical results. Finally, it will be shown that NS (Nested Structure) digital filters produce the best response for the least amount of sensitivity.

FILMED

4-84